Composición en una Sociedad de Músicos

por

Alberto Alejandro Vázquez Cortés

Tesis para obtener el grado de: Maestro en Optimización

Universidad Autónoma Metropolitana - Azcapotzalco

Posgrado en Optimización

13 de Diciembre del 2016

Asesores: . . . . . .

Dr. Roman Anselmo Mora Gutiérrez

Dr. Antonin Ponsich

.

Composición en una Sociedad de Músicos

por

Alberto Alejandro Vázquez Cortés

Resumen

En este trabajo, se presenta una nueva metaheurística denominada "Composición en una Sociedad de Músicos" (CSM); la cual basa sus ideas sociológicas sobre el comportamiento colaborativo en el Método de Composición Musical (MMC); CSM incluye además dos nuevos comportamientos sociales: aislamiento y abuso. Por otro lado se incluye el paradigma de reactividad (resultado del aprendizaje, auto-adaptación y toma de decisiones) en las metaheurísticas.

Con base en los resultados númericos obtenidos, se puede decir que la CSM ha demostrado tener una buena capacidad de resolver instancias del problema de optimización global continua sin restricciones en comparación con otras técnicas del estado del arte.

Agradecimientos

Dedicado a Jocundo Cortés y Leonida Martínez, extraordinarios seres humanos y ejemplos de vida.

Agradecimiento a mis padres que siempre han apoyado mis decisiones.

En especial, agradecimiento al Conacyt por el apoyo económico otorgado mientras realicé los estudios de posgrado; sin su apoyo nada de esto hubiera sido posible.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La optimización ha sido un área fértil de investigación por varias décadas, ya que día a día los investigadores se enfrentan a problemas de decisión u optimización cada vez más complejos. Por ende, surge la necesidad de generar, desarrollar, adaptar y/o modificar los métodos y técnicas de solución disponibles con el fin de alcanzar mejores formas de solucionar los problemas de interés.

En el presente trabajo, se aborda el problema de optimización global sin restricciones. El problema de optimización global es uno de los problemas que atrae el mayor interés de la comunidad científica y de los tomadores de decisiones, puesto que éste tiene diversas aplicaciones tales como: procesamiento de imágenes, procesos químicos y biología molecular entre otras. En la ecuación (1.1) se muestra el modelo general del problema de optimización global.

$$\begin{gathered} \text{mín}_{x \in F \subset S} f(x) \\ f : R^n \to \mathbb{R} \\ x \in \mathbb{R}^n \end{gathered} \tag{1.1}$$

Donde: x es el conjunto de las n variables de decisión; f es la función objetivo; F es la región factible y S la región de búsqueda (F y S son iguales al ser un problema sin restricciones).

Para resolver el problema de optimización global, se han propuesto una gran variedad de estrategias, dichas estrategias se pueden clasificar en procedimientos exactos, como ejemplos: método de Newton, métodos de cotas y estrategias de aproximación, tales como el recocido simulado, algoritmos genéticos, búsqueda armónica entre otros.

En el presente trabajo, se presenta una metaheurística denominada "Composición en una Sociedad de Músicos" (CSM), la cual retoma algunas ideas propuestas en el MMC (Método de Composición Musical, metaheurística desarrollada en UAM-A, de la cual se habla más adelante). Sin embargo, las principales diferencias son:

a) CSM utiliza la idea de búsqueda reactiva para el control de parámetros.

b) Los músicos en CSM poseen diferentes conductas (músicos ermitaños, músicos cooperativos y músicos parásitos), similar a los comportamietos posibles de cada ser humano en la sociedad de acuerdo a lo planteado en [21].

La estructura del presente trabajo es la siguiente:

- En la sección 2, se presenta un estado del arte sobre reactividad, comportamientos sociales y sobre las metaheurísticas empleadas internamente en CSM.
- En la sección 3, se presenta una descripción detallada de CSM.
- En la sección 4, se presenta la técnica de calibración de parámetros empleada (basada en Búsqueda Armónica).

- En la sección 5, se describen las instancias empleadas para probar la eficacia de CSM.
- En la sección 6, se presenta el procedimiento experimental, donde se describe la parte técnica del trabajo, es decir, cómo se programó la metaheurística, ejecución, requerimientos, archivos de entrada y salida, entre otros.
- En la sección 7, se presentan los resultados númericos y un análisis de los mismos.
- En la sección 8, se presentan las conclusiones y se proponen perspectivas para trabajos futuros.

Capítulo 2

Estado del Arte

2.1. Comportamientos sociales

El término sociedad proviene del latín *sociĕtas* lo que significa: compañerismo, asociación, unión. Generalmente, se utiliza el término sociedad para referirse a cualquier asociación o grupo de seres vivos, con ciertas semejanzas o coincidencias en su constitución o en sus actividades, que interaccionan entre sí y con su medio ambiente. Debe hacerse notar que la unidad resultante de la unión no elimina las diferencias entre sus integrantes, sino que éstos preservan su individualidad de seres humanos interrelacionados a través de vínculos. Los miembros de la sociedad interactúan entre sí y con su medio, para alcanzar un bien común. Los vínculos permiten: el intercambio incesante de pensamientos, ideas, conocimientos, afectos y sentimientos; y además, una constante comunicación sobre beneficios, servicios, necesidades, entre otros [18].

La sociedad humana posee y reproduce muchas de las características de la sociedad animal, como son cooperación, especialización, continuidad, entre otras; sin embargo, existe una diferencia entre las sociedades animal y la humana, la cual es la "cultura"[19]. La palabra cultura proviene del latín *cultura* que significa cultivo. En la actualidad, esta palabra se utiliza en una amplia variedad de situaciones (con diferentes significados) que van desde utilizarla para referirse a un punto de estatus o distinción; hasta aplicarla en sentido artístico para referirse a alguna disciplina artística o a los productos emanados de ella. En sociología y áreas afines, se emplea el término cultura para referirse al conjunto de tradiciones, usos y costumbres de un grupo social, etc [20]. Cultura es un término complejo que comprende conocimiento, arte, creencia, moral, usos y costumbres adquiridas por el hombre en cuanto es miembro de una sociedad; es decir, la cultura surge de la interacción social y consiste en contenidos de conocimiento y pautas de conducta que han sido socialmente aprendidas [19].

Las sociedades humanas han sido analizadas por varias disciplinas científicas, como son: sociología, medicina, inteligencia artificial, entre otras; lo que ha llevado a generar un gran número de ideas, conceptos, modelos y técnicas para su estudio. Una de estas ideas es la de red social, desarrollada desde la segunda mitad del siglo XX, como una amplicación de la teoría moderna de la comunicación al tejido de interacciones que se configura alrededor de las personas. Esta idea se basa en un enfoque multidisciplinario, el cual analiza las estruturas sociales a través de la teoría de graficas. En términos generales, una red social es un modelo que mapea todos los lazos relevantes entre un conjunto de actores (personas, organizaciones u otras entidades sociales), denominados nodos o miembros de la red, conectados por una o varias relaciones (amistad, parentesco, intereses comunes, entre otras), llamadas vínculos o aristas. Chritakis y Fowler [21] mencionan que una red social posee las siguientes normas generales: a) Se organizan y reorganizan continuamente la estructura de dicha red b) Se ven influenciados por los vínculos e interacciones con otros agentes; además, las redes sociales poseen propiedades y funciones que sus miembros no poseen, no controlan ni perciben.

Una particularidad interesante presente tanto a nivel individual como en colectivo, es la creatividad. Creatividad es el proceso inteligente para unir, juntar, asociar, conectar, integrar o combinar diferentes ideas ya existentes (previamente no relacionadas) de manera no habitual, inesperada, sorpresiva e innovadora, a fin de producir nuevas ideas que serán más complejas y mejor adaptadas para algunos propósitos [22]. La creatividad es resultado de un momento de inspiración (instante de claridad inexplicable e incomprensible); o bien, es causada por un proceso recursivo de estudio, análisis y adaptación sobre alguna idea o pensamiento [23]. La creatividad socio-cultural, también denominada creatividad colectiva, es un proceso cíclico de interacción y comunicación entre los individuos de un grupo para la construcción de conocimiento. En otras palabras, la creatividad social es un proceso de evaluación, filtrado, aceptación y aprendizaje social del conocimiento para la solución de problemas; dicho proceso es de carácter alocéntrico, ético y constructivo.

En la actualidad, se han generado y utilizado varios algoritmos de creatividad artificial para la generación de elementos artísticos en la música, en la literatura y las artes visuales. Un método es considerado como un algoritmo de creatividad artificial si parte de las siguientes premisas esenciales: a) el modelo contiene una sociedad de agentes situados en un entorno cultural; b) ningún agente puede afectar directamente el comportamiento de otro; c) existen reglas que dirigen el comportamiento global del sistema; d) los agentes interactúan con otros agentes para intercambiar ideas, artefactos y opiniones; e) los agentes interactúan con el medio ambiente para acceder a los símbolos culturales; f) los agentes pueden evaluar la calidad de los artefactos de los agentes con los que tienen un vínculo y decidir si la información o artefactos le son útiles.

Hablando de sociedades creativas en donde se ven involucrados elementos culturales, un rasgo cultural presente en casi todas las sociedades es la música. La música está presente en la vida cotidiana, ya sea, en la recreación, en la comunicación, en el arte, en la cultura o simplemente dispersa en el medio circundante [24]. La música tiene una gran influencia en el ser humano; lo cual, la ha colocado como un elemento de identidad personal y social; ya que, el sonido impacta en la conducta individual y colectiva, además, es una de las formas más antiguas de expresión [25]. La música es el arte, la ciencia y el lenguaje que organiza motivos (unidades musicales), considerando los principios de melodía, ritmo y armonía a fin de generar y trasmitir un mensaje entre el compositor y el espectador.

Un grupo de compositores dedicado a crear bellas melodías, está conformado por individuos que emplean diferentes técnicas, estilos e instrumentos musicales durante el proceso de composición, así mismo no todos los compositores se comportan de igual forma al realizar el trabajo. Su estilo musical se define con base en las experiencias obtenidas a lo largo de su trayectoria musical. Al enfrentarse a un nuevo reto, recuerdan y aprenden de experiencias previas, adaptando su trabajo musical con base en los requerimientos que la nueva melodía demande y la tendencia marcada por la sociedad. En una sociedad musical, se pueden adoptar tres comportamientos, los cuales son: cooperativos, aislados o bien parásitos. Adoptar cualquiera de los comportamientos depende de la personalidad, cosmovisión personal, estado de ánimo, valores morales, entre otros.

A continuación, se describen tres tipos diferentes de agentes inmiscuidos en el sistema sociocreativo para crear música, los cuales son:

a) *Colaborativos*. Son aquellos que deciden ser productivos asociándose con otros, dividen el trabajo y comparten tanto su talento como su conocimiento.

b) *Aislados*. Son aquellos que se enfocan en trabajar sin tomar en cuenta la obra de otros, concentrándose en su propio talento y experiencias.

c) *Parásitos*. Son aquellos que se aprovechan del trabajo de los demás, compilando las mejores melodías en la sociedad, pero se niegan a compartir su conocimiento o talento con los

demás. Estos compositores producen nuevos elementos recombinando la mejor información que han "robado", lo cual permite buscar nuevos trabajos en regiones prometedoras acotada por información de élite.

Además, del comportamiento social, los individuos presentan elementos propios que influyen en su comportamiento y toma de desiciones. En estos procesos internos se implican los mecanismos de aprendizaje basados en experiencia y adaptación.

2.2. Reactividad

En los seres humanos los procesos de resolución de problemas se encuentran fuertemente ligados al aprendizaje adquirido a lo largo de la vida. En términos generales, el aprendizaje se puede ver como el conocimiento, o la experiencia, adquirido por la interacción de las neuronas al procesar la información referente a los estímulos externos involucrados en el proceso de resoclución (fase que supone la conclusión de un proceso más amplio que tiene como pasos previos la identificación del problema y su modelado). Cabe mencionar, que cada persona puede acceder, modificar y usar su conocimiento en la toma de decisiones, ésta es la idea base para el paradigma de búsqueda reactiva.

El concepto de reactividad se ha empleado en diversas metaheurísticas desde hace tiempo. Uno de los aspectos donde más se ha empleado es en el control de parámetros. La diferencia entre calibración (configuración) de parámetros y control de parámetros es que la calibración se ejecuta fuera de línea, es decir, antes de la ejecución de la metaheurística; mientras que el control lo realiza la propia metaheurística internamente en cada iteración. El valor que se le asigne al parámetro en cuestión, depende de las circunstancias, escenario o tiempo en el que se encuentra la ejecución de la metaheurística [26]. Por ejemplo, Recocido Simulado (SA), simula el proceso de recocido del acero, en donde inicialmente la temperatura del acero es muy alta y luego sucede un enfriamiento progresivo y paulatino hasta un valor muy bajo, con la finalidad de que los átomos internos se acomoden de forma óptima y con esto conseguir un acero de óptima calidad. En una versión reactiva, la temperatura se controla no sólo considerando el principio de que se tiene que llegar a 0 (o un valor muy bajo), sino que se puede aumentar, o disminuir de golpe con base en las circunstancias de la búsqueda en ese momento. Estos nuevos principios o reglas toman en consideración elementos de memoria, es decir, se aumenta la temperatura o se disminuye tomando como referencia si esto funcionó bien anteriormente. El momento en que se toma la decisión de: *a)* aumentar la temperatura de golpe, *b)* continuar con un decremento paulatino o *c)* bajarla de golpe, sucede cuando la búsqueda se estanca sin conseguir mejores resultados durante repetidas iteraciones. En ese momento se tienen tres posibles decisiones, y la decisión se toma considerando el temperatura actual, es decir el escenario o circunstancia, y por otro lado considerando los casos de éxito que han tenido cada decisión, es decir, experiencias previas.

Otro aspecto en donde se ha implementado la reactividad dentro de una metaheurística es en el problema de dirigir la búsqueda hacia una dirección u otra(s). Esta decisión se toma con base en las experiencias previas, es decir, recordando que es lo que ha funcionado con mejores resultados en el pasado. Durante la ejecución, la metaheurística se va a enfrentar a esta decisión en repetidas ocasiones; si se guarda esta información, mas el resultado obtenido (si fue beneficioso o no), se tendrá una memoria o base de datos que ayudará a tomar la mejor decisión en las iteraciones posteriores.

Queda claro que para que una metaheurśtica sea reactiva, debe de tener una memoria y un conjunto de reglas que son ejecutadas cuando se toma una decisión que impacta en el proceso de búsqueda. La decisión que se tome es influenciada por las experiencias previas y las mejores decisiones tienen mayores probabilidades de ser elegidas.

Ahora bien, en una metaheurística poblacional, en donde están involucrados varios agentes o individuos, existe la posibilidad de que cada agente tenga su propia memoria y sus propias reglas para tomar decisiones. O bien, compartir memoria, compartir las reglas o ambas. Si la memoria es grupal, todos los agentes tendrán acceso a ella y todos aportarán sus experiencias, lo cual enriquece la base de datos considerando diferentes perspectivas.

A lo largo de los años, se han introducido diferentes conceptos referentes a la relación entre reactividad y sistemas multi-agentes. Bredenfeld y Kobialka [5] presentan un enfoque basado en la coordinación grupal, el cual es una extensión del esquema basado en comportamientos duo-dinámicos; en dicho trabajo se introducen "variables grupales"que son leídas y actualizadas por los agentes de forma reactiva. Este enfoque fue aplicado para coordinar robots en un equipo de futbol soccer. Por otro lado, Behnke y Rojas introducen en 2001 [6], "jerarquías de comportamientos"; rápido y simple al comienzo de la jerarquía y comportándose más lento y complejo conforme se llega al final. El sensor de modulación se ajusta de forma reactiva con base en las características temporales del sistema. Este enfoque fue aplicado en RoboCup, un torneo de futbol soccer entre robots (http://www.robocup.org).

2.3. Metaheurísticas como técnicas para resolver problemas de optimización

Se distinguen comúnmente dos grandes clases de técnicas de optimización, por un lado se encuentran los métodos exactos que están basados en un estudio matemático riguroso de las funciones involucradas en el modelo, lo que puede implicar dificultades serias, relacionadas con la no-linealidad y la no-continuidad de las funciones. Por otro lado, se encuentran las metaheurísticas que pueden ser vistas como heurísticas de alto nivel, diseñadas para encontrar, generar o seleccionar una solución lo suficientemente buena a un problema de optimización, especialmente cuando no se cuenta con toda la información o se tiene capacidad computacional limitada. Es importante mencionar que las metaheurísticas representan una buena alternativa para tratar los problemas NP-completos y NP-duros.

Hoy en día, se han desarrollado un gran número de metaheurísticas. Algunas de ellas intentan reproducir fenómenos presentes en la naturaleza, como las técnicas de Cómputo Evolutivo, sin dejar de mencionar otros enfoques como el de Búsqueda local y las metaheurísticas basadas en trayectorias [15].

El paradigma de las técnicas de Cómputo Evolutivo data de principio de los años 50's cuando surge la idea de usar los principios darwinianos [8] con el propósito de solucionar problemas relacionados con autómatas. No fue hasta los 60's, que tres distintas interpretaciones de esta idea comenzaron a desarrollarse en tres diferentes lugares. Programación Evolutiva (EP) fue introducida por Lawrence J. Fogel [7] en los Estados Unidos, y casi simultáneamente Rechenberg y Schwefel introducen Estrategias Evolutivas (ESs) en Alemania en 1964. Casi una década después, John Henry Holland de la Universidad de Michigan crea un método que simula la teoría de la evolución darwiniana para resolver problemas de optimización, llamado Algoritmo Genético (GA) [9]. A principio de los 90's se unifican en una sola rama de estudio llamada "Cómputo Evolutivo". Asimismo, al principio de los 90's una nueva ideología inspirada en las tres anteriores emerge, llamada Programación Genética (GP).

Hoy en día, el campo de las Metaheurísticas inspiradas en la naturaleza, en su gran parte está constituido por Algoritmos Evolutivos (incluidos GAs, EP, ESs, GP, Evolución diferencial - ED [3], entre otros), los algoritmos de Inteligencia Colectiva, por ejemplo: Optimización por enjambre de partículas (PSO) [10], Optimización por colonia de hormigas (ACO)[11], Optimización por Búsqueda con Bacterias (BFO) [12], etc. También, el campo se extiende en un sentido más amplio para incluir los Algoritmos Meméticos (MAs), Algoritmos Culturales, Búsqueda

Armónica (HS) [4] entre otros.

De estas nuevas vertientes cabe recalcar a PSO y ED como técnicas de reciente creación, que han sido foco de investigación en las últimas décadas, en gran medida por los buenos resultados obtenidos al momento de resolver problemas de optimización continua.

2.3.1. Metaheurísticas basadas en sociedades

Las metaheurísticas basadas en sociedades pueden ser vistas como un subconjunto de los métodos evolutivos. En los procedimientos que imitan sociedades, la búsqueda es influenciada por cambios en el entorno social, así como por interacción entre los individuos (agentes) y con su entorno; en otras palabras, los individuos son entidades fundamentalmente sociales que interactúan entre si con el propósito de mejorar. Dichas mejoras se alcanzan: por las aportaciones que hacen cada uno de los individuos en la sociedad, las interacciones que inducen procesos de aprendizaje entre los agentes y a la adaptación veloz de los agentes a su entorno. La inspiración de muchas de estas técnicas proviene de procesos naturales observados en sistemas biológicos y/o sociales.

La técnica de Composición en una Sociedad de Músicos (CSM), desarrollada en este trabajo, emplea internamente cinco metaheurísticas con el propósito de generar nuevas soluciones con base en mecanismos reactivos. Se han incluido dentro del procedimiento porque han sido implementadas para resolver numerosas instancias del problema de optimización global con buenos resultados. Cada uno de los diferentes métodos de búsqueda realiza un balance entre diversificación e intensificación, apoyándose sobre elementos de memoria, experiencia, colaboración, adaptación, selección, entre otras técnicas para dirigir la búsqueda, salir de óptimos locales e intensificar la búsqueda en zonas prometedoras.

A continuación, se describen las metaheurísticas empleadas por CSM para generar soluciones. Es importante mencionar que CSM no utiliza el procedimiento original al pie de la letra, sino que emplea una adaptación, la cual será explicada posteriormente. Dicha adaptación se debe a que las metaheurísticas seleccionadas generan en cada iteración más de una solución (a excepción de HS); en cambio, CSM sólo necesita una. La adaptación es llamada como "Metaheurística generadora de soluciones".

2.3.2. Colonia de Abejas Artificiales

Dentro de los algoritmos inspirados en la naturaleza para resolver problemas de optimización, están los basados en inteligencia colectiva (Swarm Intelligence), que son aquellos que se enfocan en el comportamiento colectivo, el control descentralizado y la auto-organización. Dentro de estos algoritmos podemos encontrar a la colonia de hormigas (ACO), bancos de peces (AFSA) [13], parvadas de pajáros y la colonia de abejas artificiales (ABC). ABC fue propuesto en 2005 por Karaboga [2] y está basado en el forrajeo (recolección de polen y nectar) de las abejas melíferas.

ABC imita dos aspectos importantes que realizan las abejas cuando buscan su alimento:

- Búsqueda local (a la cercanía).
- Búsqueda dirigida (hacia la mejor fuente de alimento).
- Exploración (visitar regiones nunca antes exploradas).

En este proceso interactúan tres tipos de abejas, las cuales son:

a) Abeja supervisora: Está al mando y dirige la búsqueda, tiene una abeja obrera a su cargo para realizar las búsquedas. Conoce la ubicación de la mejor fuente de alimento (mejor solución) encontrada hasta el momento.

b) Abeja obrera: Realiza la búsqueda local y/o dirigida con base en lo que su supervisora le ordene.

c) Abeja Exploradora: Busca nuevos puntos de partida (aleatorios) para explorar el espacio de búsqueda.

La idea es la siguiente: Se realiza una búsqueda en la cercanía y posteriormente una búsqueda en dirección a la mejor fuente de alimento. Si las dos estrategias anteriores no logran encontrar una fuente de alimento mejor que la fuente de alimento inicial, entonces se explora el paisaje de búsqueda desde otra perspectiva o posición. A lo largo de la búsqueda, las abejas se posicionan en puntos donde la fuente de alimento es buena, siempre dirigiendo la búsqueda en dirección a la mejor fuente de alimento. La diversificación se hace presente cuando la abeja explora nuevos territorios, lo que ayuda también a escapar de óptimos locales.

2.3.3. Algoritmo Genético

Un investigador de la Universidad de Michigan llamado John Holland era consciente de la importancia de la selección natural, y a fines de los 60's desarrolló una técnica que permitió incorporarla a un programa. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "Algoritmo Genético"tras la publicación de su libro en 1975 [9].

En la literatura se han reportado aplicaciones de los algoritmos genéticos (AG) para una gran variedad de problemas de optimización. Esta metaheurística se inspira en el proceso de evolución de individuos de una población, donde los individuos mejor adaptados tienen mayores probabilidades de sobrevivir y reproducirse.

A continuación, se describen las fases del proceso de búsqueda:

- Selección de individuos para cruza: Por cada individuo a generar (descendiente) se seleccionan dos individuos padres, diferentes entre si; se les asigna una mayor probabilidad de ser elegidos a los mejores individuos, sin descartar la selección estocástica.
- Cruza: Se aplica un mecanismo de cruza para generar los nuevos individuos. El mecanismo de cruza debe proveer al nuevo individuo características de ambos padres.
- Mutación: Bajo cierta probabilidad, típicamente baja, se realiza la mutación de los nuevos individuos, que consiste en alterar aleatoriamente la nueva solución en un rango corto.
- Reemplazo: Se remplazan individuos de la generación anterior por los nuevos individuos. Los mejores individuos de la generación anterior sobreviven a la próxima generación.

2.3.4. Búsqueda Armónica

Búsqueda Armónica (HS) fue inicialmente desarrollada por Zong Woo Geem et al. en 2001 [4]. Al pasar de los años, su eficiencia y sus ventajas han sido demostradas en varias aplicaciones; desde entonces, ha sido implementada para resolver muchos problemas de optimización incluyendo optimización global, flujos en redes, modelado de flujos acuíferos subterráneos, sistemas de ahorro de energía, ruteo de vehículos, entre otros.

HS es una metaheurística basada en la analogía del comportamiento de un músico al crear o imporovisar música; el músico busca un perfecto estado de armonía en las notas. La armonía en la música es equivalente a encontrar la solución óptima en un proceso de optimización. El proceso de búsqueda en optimización puede ser comparado con el proceso de improvisación de

un músico de jazz. Por otra parte, la armonía es determinada por el estándar de belleza del sonido musical.

Búsqueda Armónica es el perfecto ejemplo, en donde se transforma el proceso cualitativo de improvisación, en un conjunto de reglas cuantitativas. Así pasamos de la belleza de la armonía en la música, a un proceso de optimización que busca la perfecta armonía.

Procedimiento. Cuando un músico está improvisando, se tienen tres posibles decisiones: (1) tocar una famosa pieza de música en su memoria (2) tocar algo similar a una pieza que recuerde (3) componer algo nuevo y ser creativo. Formalizando estas tres componentes en un algoritmo, se integran a la metaheurística como: uso de la memoria armónica, ajuste de tono y aleatoriedad.

El uso de la *memoria armónica* es importante, similar a la elección de los mejores individuos en el algoritmo genético. Esto asegura que las mejores armonías continúen en la nueva memoria armónica. El parámetro $r_{aceptacion}$ [0,1] usualmente llamado "aceptación de memoria armónica" controla el uso de la memoria armónica al momento de crear nuevas armonías. Si el valor asignado es muy pequeño, sólo pocas buenas armonías se seleccionan, lo que posiblemente provoque que el algoritmo converja lentamente. Si el valor es exageradamente alto (cercano a 1), la gran mayoría de las armonías se toman de la memoria armónica, lo que provoca que no se exploren otros valores que podrían dar mejores resultados. Por lo tanto, típicamente se asigna un valor entre $r_{aceptacion}$= [0.7, 0.95].

El segundo componente es el *ajuste de tono*, determinado por el ancho de banda b_{rango} y el parámetro de ajuste de tono r_{ajuste}. Como en la música, el ajuste de tono significa el cambio en alguna frecuencia, que corresponde a generar una solución ligeramente diferente. El ajuste utilizado es el siguiente:

$$X_{nueva=}X_{vieja+}b_{rango}$$

donde X_{vieja} es el tono o solución existente de la memoria armónica, y X_{nueva} es el nuevo tono después del ajuste de tono. En esencia esta operación produce una nueva solución basada en algo que ya teníamos con una ligera perturbación o cambio de tono, típicamente en un rango pequeño. El ajuste de tono es similar a la mutación si lo comparamos con el algoritmo genético.

Se puede controlar la frecuencia del ajuste de tono de una armonía mediante el parámetro r_{ajuste}. Un pequeño cambio en el tono en un ancho de banda estrecho puede hacer más lenta la convergencia, ya que limitamos la exploración a un pequeño sub-espacio del espacio de búsqueda. En el sentido opuesto, un gran cambio en el tono en un ancho de banda grande podría causar que la búsqueda se torne un tanto aleatoria. Usualmente se toma r_{ajuste}= 0.5 en la mayoría de las aplicaciones.

El tercer componente es la *aleatoriedad*, que tiene como finalidad incrementar la diversidad en las soluciones. Aunque el ajuste de tono tiene un propósito similar, éste está limitado a un ancho de banda o espacio de movimiento definido. El uso de la aleatoriedad puede conducir a espacios inexplorados donde se puede encontrar el óptimo global.

A continuación, se presenta el algoritmo canónico de HS. Cabe mencionar que HS sólo genera una solución en cada iteración.

Algoritmo 1 HS

Entrada:

0: $r_{aceptacion}$ Rango de aceptación de la memoria armónica.
0: r_{ajuste} Frecuencia de ajuste de tono.
0: b_{rango} Ancho de Banda.
1: Generar un conjunto inicial de n arreglos armónicos (soluciones). Evaluar cada solución con respecto a la función objetivo e integrarla a la memoria armónica.
2: **mientras** no se cumple un criterio de paro **hacer**
3: Se genera un arreglo armónico vacío. $x = (x_1, x_2, x_3, \ldots, x_d)$
4: **para** cada variable i del vector solución (desde *1* hasta d) **hacer**
5: **si** aleatorio1 $< r_{aceptacion}$ **entonces**
6: Aceptación de la memoria armónica (Recordar). Se elige de manera aleatoria un arreglo armónico de la memoria armónica y se toma el valor de la variable i para asignarlo en el valor de la variable i del nuevo arreglo armónico.
7: **si no**
8: **si** aleatorio2 $< r_{ajuste}$ **entonces**
9: Ajuste de tono en ancho de banda (Generar algo ligeramente diferente). Se elige de manera aleatoria un arreglo armónico de la memoria armónica, se toma el valor de la variable i y se ajusta el valor de manera aleatoria dentro del ancho de banda (b_{rango}). Finalmente se asigna al valor de la variable i del nuevo arreglo armónico.
10: **si no**
11: Aleatoriedad (Creatividad). Generar una nueva armonía totalmente aleatoria dentro del espacio de búsqueda y se asigna al valor de la variable i del nuevo arreglo armónico.
12: **fin si**
13: **fin si**
14: **fin para**
15: Evaluar a x con respecto a la función objetivo.
16: **si** x mejor que el peor arreglo armónico **entonces**
17: Remplazar al peor arreglo armónico por x.
18: **si no**
19: Desechar x.
20: **fin si**
21: **fin mientras**
22: **devolver** Mejor arreglo armónico (solución)

La figura 2-1, ilustra los tres movimientos posibles en HS: memoria armónica, ajuste de tono y aleatoridad.

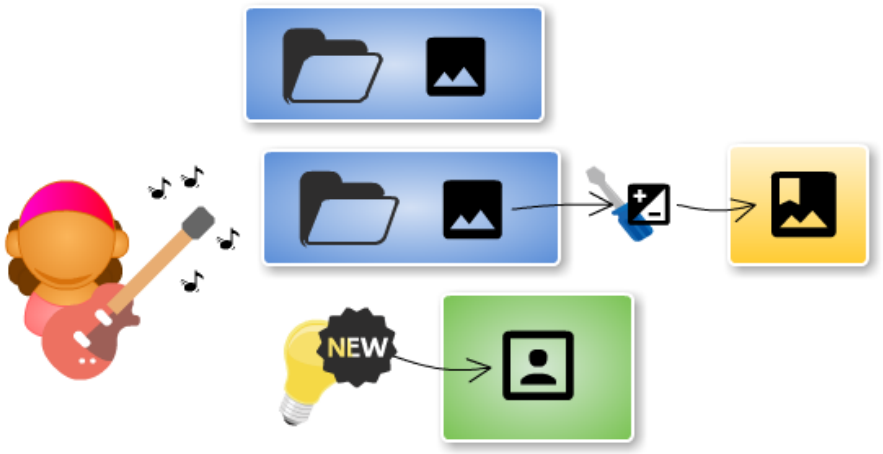

Figura 2-1: Memoria armónica (recordar), ajuste de tono (recordar y modificarlo un poco) y aleatoridad (creatividad)

2.3.5. Evolución diferencial

El algoritmo de Evolución diferencial (ED) fue introducido por Storn y Price en 1995 [3], dicho algoritmo es un Algoritmo Evolutivo diseñado para trabajar con variables reales, involucrando un mecanismo novedoso para generar una solución candidata y un esquema de reemplazo glotón. ED trabaja de la siguiente manera: Inicialmente todos los individuos son creados de forma aleatoria. Después, por cada individuo en la población, se crea un descendiente, el cual es creado realizando operaciones entre vectores, combinando direcciones y magnitudes, o manteniendo la dirección pero aumentando o disminuyendo la magnitud. El descendiente reemplaza al padre sólo si este es mejor, de otra forma el padre sobrevive a la siguiente generación.

A continuación, se describen las cuatro etapas del algoritmo de ED:

1. Inicialización: Inicializar los individuos de la población de forma aleatoria considerando como posibles valores, todos los definidos por el espacio de búsqueda.

2. Mutación: Sea a la solución de partida o "Individuo original", y P la población, donde $a \in P$. Se deben elegir dos elementos de $P : b, c$ tal que $b \neq c, a \neq b, a \neq c$, es decir que sean elementos distintos entre si; se debe de cumplir que $|P| \geq 3$. Se genera una nueva solución x, realizando la siguiente operación, tomando a las soluciones a, b, c como vectores: $x = a + (Pctrl)(b - c)$, donde $Pctrl$ es un valor real entre 0 y 1. La nueva solución x, se conocerá como "Mutante".

3. Recombinación: Se hace uso de la técnica "Cruza exponencial", en donde se combinan elementos (variables) del "Individuo original" con su "Mutante", con base en el factor de cruza que determina qué tantas características mutantes se le otorgan al "Nuevo individuo". Es decir, la solución resultante o "Nuevo individuo" tiene tanto sus características originales (antes de la mutación), como las de su mutante con base en el factor de cruza.

4. Selección: Si el individuo resultante de la recombinación es mejor que el individuo original, se remplaza en la población.

En cada generación se repiten los pasos 1, 2 y 3 para cada individuo de la población.

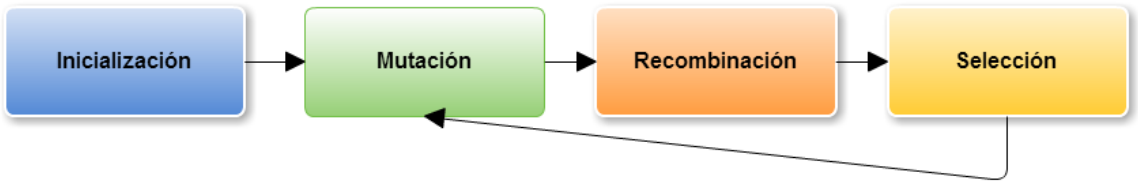

Figura 2-2: Procedimiento de Evolución Diferencial

2.3.6. Búsqueda Dispersa

Los conceptos y principios fundamentales de Búsqueda Dispersa (SS) [14] fueron propuestos a comienzo de la década de los setenta. La primera descripción del método fue publicada en 1977 por Fred Glover donde establece los principios de SS.

En una comparación con AG, esté mantiene un conjunto de soluciones relativamente grande, selecciona las mejores de manera probabilística, realizando operaciones de recombinación y mutación entre ellas para formar la siguiente generación. En cambio, en SS la selección de soluciones es sistemática y estratégica sobre un conjunto pequeño. Como ilustración basta decir que los algoritmos genéticos suelen considerar una población de 100 soluciones mientras que en la búsqueda dispersa es habitual trabajar con un conjunto de tan sólo 10 soluciones.

A continuación, se describen los pasos del algoritmo canónico:

1. Generación de soluciones diversas: El método genera un conjunto P de soluciones diversas (alrededor de 100), del que se extrae un subconjunto pequeño (alrededor de b=10) con el que se realizan las combinaciones y que se denomina como: conjunto de referencia.

2. Conjunto de referencia (*Ref-Set*): Extraído del conjunto de las soluciones diversas según dos criterios: (1) contener soluciones de calidad (2) que sean diferentes entre si (calidad y diversidad). Las soluciones en este conjunto están ordenadas de mejor a peor respecto de su calidad.

   a) Creación: Se inicializa el conjunto de referencia con las $b/2$ mejores soluciones de P. Las $b/2$ restantes se extraen de P por el criterio de máxima distancia con las ya incluidas en el conjunto de referencia. Para ello se debe definir previamente una función de distancia.

   b) Actualización: Las soluciones fruto de etapas de combinación y mejora, pueden entrar en el conjunto de referencia y reemplazar a alguna de las ya incluidas si las mejoran. Así pues, el conjunto de referencia mantiene un tamaño b constante pero va mejorando a lo largo de la búsqueda.

3. Método de combinación: se combinan las soluciones del conjunto de referencia. Para ello, se consideran subconjuntos de dos o más elementos de dicho conjunto y se combinan mediante un método de cruza.

4. Método de mejora. Típicamente se trata de un método de búsqueda local para mejorar las soluciones, tanto del conjunto de referencia como de las resultantes de la combinación, antes de considerar su inclusión en el conjunto de referencia.

2.3.7. Método de Composición Musical

El Método de Composición Musical (MMC) fue propuesto por el Dr. Román Anselmo Mora y el Dr. Javier Ramírez Rodríguez (entre otros)[17], para resolver problemas de optimización

no restringida. Este algoritmo socio-cultural está basado en la evolución de una sociedad artificial de agentes (o compositores), que tienen una capacidad creativa propia y que pueden intercambiar información entre sí. Esta sociedad es dinámica, por lo que los vínculos entre los diferentes agentes pueden crearse o desaparecer en el transcurso de la evolución. Los agentes son libres de utilizar o no la información de los demás agentes con los cuales están en contacto. El comportamiento de intensificación adoptado por los agentes (cada agente busca en su "entorno") se ve balanceado por la capacidad exploradora debida a la interacción entre ellos. A continuación, se describe brevemente el modo operativo del algoritmo:

1. Se crea una sociedad de compositores, con vínculos entre ellos generados de forma aleatoria. Cada compositor tiene un conjunto inicial de melodías (soluciones), también generadas de forma aleatoria. Este conjunto de melodías forma el *conocimiento propio* del compositor, su obra musical de acuerdo a la metáfora del MMC.

2. Mientras no se cumple un criterio de paro (típicamente, mientras no se alcanza un tiempo límite de ejecución o un número máximo de iteraciones), se realizan los siguientes pasos:

   a) Cada agente intercambia información con sus vecinos, aceptando o rechazando una melodía para construir su *conocimiento adquirido* .

   b) La unión de los *conocimientos propio* y *adquirido* de cada agente forma su *experiencia*.

   c) Cada agente construye una nueva melodía. Para ello, puede tomar en cuenta su *experiencia* completamente (mediante una recombinación de varias de sus melodías conocidas), parcialmente (modificando localmente una de sus melodías conocidas) o para nada (inventando una melodía completamente nueva).

   d) Cada agente decide integrar o no la melodía nueva a su conocimiento propio, generalmente de acuerdo a un criterio glotón.

   e) Se actualiza la sociedad artificial, de tal forma que cada agente puede eliminar vínculos existentes con sus vecinos y/o crear vínculos nuevos con otros agentes.

Varios experimentos numéricos han permitido comprobar que los resultados obtenidos por el algoritmo MMC sobre bancos de instancias con funciones continuas (restringidas o no) son muy satisfactorios cuando se comparan con otras técnicas metaheurísticas del estado del arte [17].

En la figura 2-3, se pueden observar los integrantes de la sociedad de compositores, así como los vínculos o asociaciones entre ellos; estos vínculos cambian con el paso del tiempo.

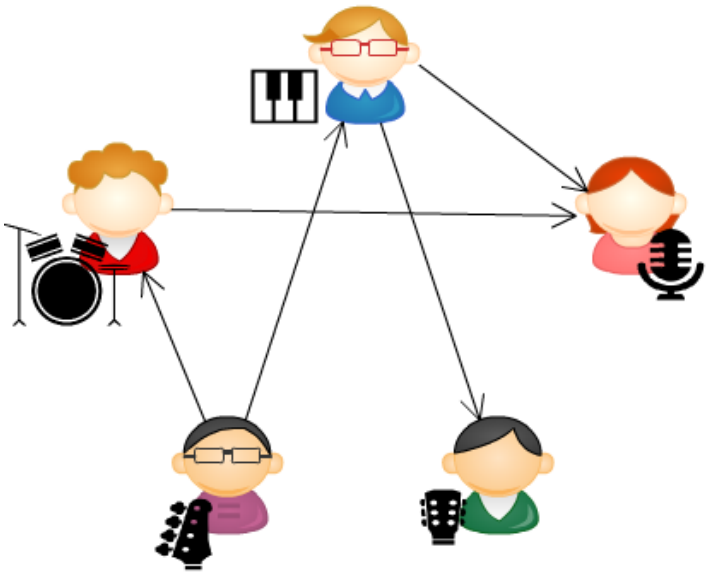

Figura 2-3: Intergrantes de la sociedad y los vínculos entre ellos (MMC)

Capítulo 3

Descripción del método desarrollado

Como antes se ha mencionado, la metaheurística "Composición en una sociedad de músicos" (CSM) presenta dos líneas de investigación:

- Comportamientos sociales,
- Reactividad.

El tipo de comportamiento adoptado determina cuáles melodías están involucradas al momento de generar una nueva solución. Por otro lado, la reactividad decide qué técnica es la más conveniente para generar nuevas soluciones con las melodías involucradas en el proceso de composición musical.

Estos dos aspectos de la metaheurística se abordarán de manera separada al explicar la metodología empleada, pero ambas conforman el algoritmo general. Elegir el comportamiento social (influenciado por los parámetros de entrada) es la parte crucial del algoritmo; por otro lado, la reactividad se emplea para imitar el comportamiento de aprendizaje, y se aplica sin importar el comportamiento del compositor (aislado, parásito o colaborador).

3.1. Comportamiento Social

Algunos músicos deciden ser productivos asociándose con otros para colaborar, dividir el trabajo y compartir su talento y su conocimiento. En este caso, la metaheurística se inspira en el MMC, reproduciendo sus mecanismos de intercambio de información y colaboración entre los individuos.

Por otro lado, se introduce otro tipo de compositor que decide trabajar en solitario, ya que así se enfoca sobre su propia obra sin tomar en cuenta la obra de los demás, concentrándose en su propio talento y experiencias.

Finalmente, un subconjunto de compositores se aprovecha del trabajo de los demás, compilando las mejores melodías producidas por la sociedad, ahorrándose con ello el trabajo de crear sus propias melodías.

Los compositores de esta última clase sólo producen melodías nuevas recombinando elementos de las mejores, garantizando de esta forma trabajos de buena calidad y con buena aceptación, pero poco originales. Esta clase de compositores permite crear soluciones nuevas en regiones intermedias de algunas soluciones élite (mejores soluciones encontradas hasta el momento), de manera similar a las técnicas de re-encadenamiento de trayectorias. El re-encadenamiento de trayectorias (Glover y Laguna, 1997) [28] se basa en el hecho de que entre dos soluciones se puede trazar un camino que las una, de modo que las soluciones en dicho camino contengan atributos de ellas.

En la figura 3-1, se ilustra la composición de la sociedad de músicos. El punto verde indica la mejor melodía de cada compositor colaborativo, que a su vez, son compiladas por el compositor parásito. El compositor aislado, realiza un balance entre sus mejores y más diversas melodías (puntos amarillos y azules) aislándose de los compositores colaborativos.

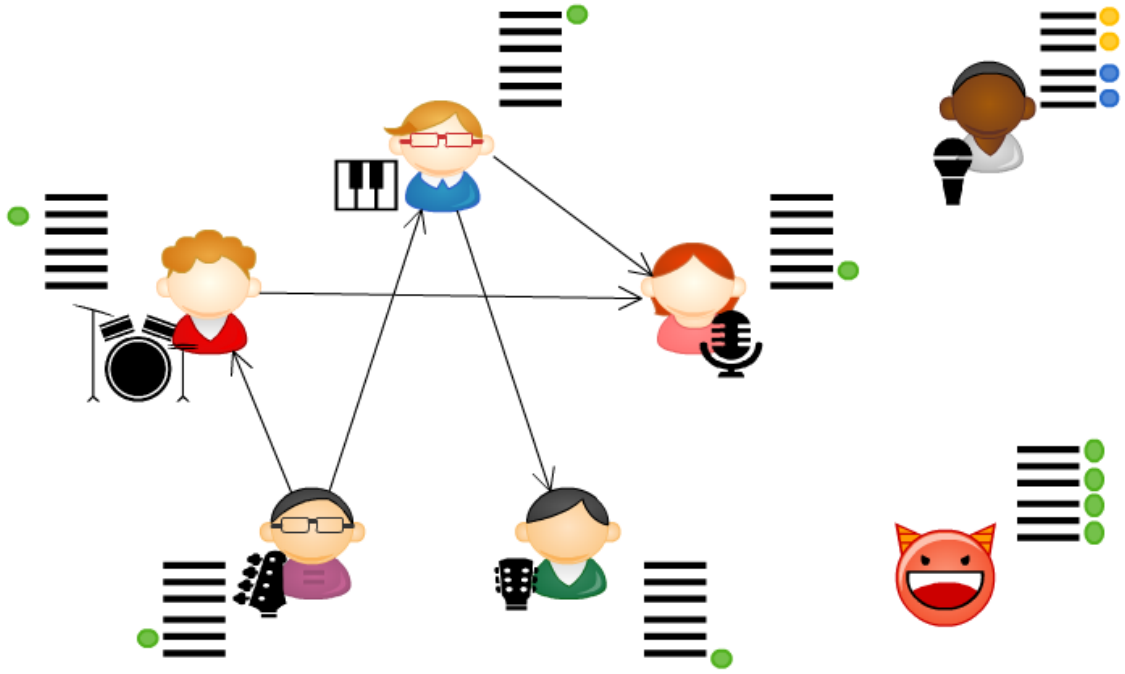

Figura 3-1: Individuos colaborativos, individuo aislado e individuo parásito

3.1.1. Parámetros de entrada

En toda metaheurística, encontrar la configuración de parámetros de entrada óptima es un requerimiento esencial para encontrar soluciones de buena calidad, en el menor número posible de iteraciones. CSM no es la excepción, la metaheurística tiene cuatro parámetros de entrada que influyen en el desempeño de la misma, involucrando los siguientes dos aspectos:

- La tendencia en el comportamiento (colaborativo, aislado, parásito) de los compositores.
- El número de compositores y su capacidad de retención de memoria al paso de las iteraciones.

A continuación, se describen los cuatro parámetros de entrada:

1. Tasa de trabajo colaborativo (tco). Valor real entre 0 y 1. Define la probabilidad de que los compositores colaboren entre sí, o en caso contrario que se comporten de forma aislada o abusiva.

2. Tasa de comportamiento parásito ($tpar$). Valor real entre 0 y 1. Define la probabilidad condicional de que, dado que no se comportan de forma colaborativa, los compositores adopten un comportamiento parásito o, al contrario, aislado.

3. Número de compositores (n): Valor entero entre 2 e ∞. Número de compositores que colaboran entre si. Sólo existe un compositor aislado y un parásito, independientemente del número de compositores colaborativos.

4. Número de melodías por compositor (m). Valor entero entre 3 e ∞. Capacidad de memoria por compositor, es decir, cuántas melodías puede recordar un agente simultaneamente a lo largo de la ejecución del algoritmo. Cabe mencionar, que en el caso del compositor parásito, el tamaño de su memoria es n.

3.1.2. Algoritmo general

Con base en lo enunciado anteriormente, acerca de los diferentes comportamientos sociales que pueden adoptar los compositores, el algoritmo 2 presenta la estructura general del método de CSM. En los renglones 3 y 8, se realizan las pruebas para decidir el tipo de comportamiento que los compositores adoptarán.

Algoritmo 2 Algoritmo CSM

Entrada: Parámetros de entrada: (*tco*, *tpar*, n, m)

```
1: Inicialización de la sociedad y agentes reactivos (n, m)
2: mientras No se cumpla un criterio de paro hacer
3:    si aleatorio1 < tco  entonces
4:       para todo Compositores colaborativos hacer
5:          Proceso de composición de compositor colaborativo
6:       fin para
7:    si no
8:       si aleatorio2 < tpar entonces
9:          Proceso de composición de compositor parásito
10:      si no
11:         Proceso de composición de compositor aislado
12:      fin si
13:   fin si
14:   Actualizar vínculos en la sociedad (compositores colaborativos), cada músico puede eli-
      minar o agregar vínculos con sus colegas.
15:   Se comparten las melodías (conocimiento adquirido).
16: fin mientras
17: devolver  Mejor melodía (solución) encontrada
```

Donde:

- aleatorio1, aleatorio2: son números reales aleatorios e independientes con distribución uniforme entre 0 y 1.
- Criterio de paro: consiste típicamente en alcanzar un tiempo límite de ejecución, un número máximo de iteraciones o un número máximo de llamadas a la función objetivo.

3.1.3. Inicialización de la sociedad

Este proceso crea los compositores y su memoria propia (melodías o soluciones) en base a los parámetros de entrada. Además, crea las relaciones entre los compositores colaborativos. A continuación, se describen los pasos a seguir:

- Crear n compositores colaborativos, cada uno con m melodías (soluciones). A cada una de las notas (variables) que integran la melodía, se le asigna un valor aleatorio uniforme entre sus cotas.

- Determinar, para cada compositor colaborativo, su mejor melodía y guardarla dentro del arreglo de mejores melodías de tamaño n. Dicho arreglo constituye la memoria del compositor parásito.

- Crear el compositor aislado con m melodías genaradas de forma distribuida. Es decir, se procura que las soluciones abarquen en lo posible todo el espacio de búsqueda.

- Crear las relaciones entre los individuos colaborativos de acuerdo a los siguientes pasos:

  - Por cada compositor colaborativo se decide si se crea un vínculo con cada uno de los otros compositores colaborativos, con una probabilidad de 0.5; es decir, existe una probabilidad igual que el vínculo sea creado o no.
  - Se impone que cada compositor tenga al menos un vínculo con otro compositor; por lo que, si no se crea ningún vínculo en el paso anterior, se elige de forma aleatoria un compositor y se crea el vínculo con el. El motivo de lo anterior es que, de no tener ninguna colaboración con otro agente, el compositor se comportaría como un compositor aislado.

3.1.4. Proceso de composición

El proceso de composición, es la parte del algoritmo en donde el compositor, con base en el conocimiento disponible, inicia la creación de nuevas melodías, mediante estrategias reactivas. Posteriormente, se decide si las melodías se integran o se desechan en la memoria del compositor, con base en una calificación obtenida. Cada tipo de compositor (colaborativo, aislado o parásito) tiene su propia forma de componer música tomando como referencia sus melodías base; entiéndase como melodías base, las melodías del conocimiento del compositor, es decir, las melodías a las que tiene acceso. A continuación, se describe, para cada compositor, su proceso de composición.

Proceso de composición para compositores colaborativos

- Número de melodías a componer: n nuevas melodías, una por cada compositor colaborativo.

- Melodías base: Se toma como base las m melodías del compositor (conocimiento propio), además de las aportadas por la sociedad (conocimiento adquirido).

- Calificación: Se dice que una melodía generada es buena si es mejor que la peor melodía del compositor.

- Reemplazo: Si la melodía fue calificada como buena, reemplaza a la peor melodía del compositor. También se valida si se puede integrar al arreglo de mejores soluciones.

En la figura 3-2, se ilustra a cada músico colaborativo componiendo su propia melodía, con base en su propio conocimiento (melodías verdes) y el aportado por la sociedad (melodías amarillas).

Figura 3-2: Compositores Colaborativos

Proceso de composición para compositor parásito

- Número de melodías a componer: n nuevas melodías.
- Melodías base: Se toman como base las n mejores melodías (arreglo de mejores soluciones).
- Calificación: Se dice que una melodía generada es buena si es mejor que la peor melodía contenida en arreglo de mejores melodías.
- Reemplazo: Si la melodía fue calificada como buena, reemplaza a la peor melodía del arreglo de mejores melodías.

En la figura 3-3, se ilustra al compositor parásito; robando las mejores melodías de los compositores colaborativos y el compositor aislado.

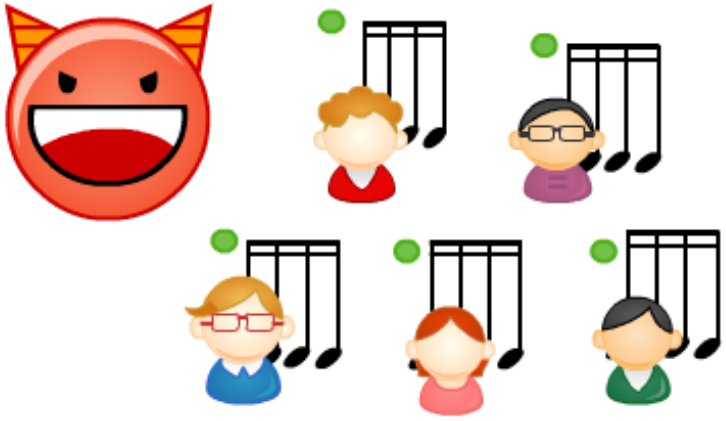

Figura 3-3: Compositor Parásito

Proceso de composición para compositor aislado

- Número de melodías a componer: m nuevas melodías.
- Melodías base: Se toma como base las m melodías del compositor aislado.

- Calificación: Se dice que una melodía generada es buena si es mejor que la peor melodía del compositor aislado.

- Reemplazo: Una vez generadas las m nuevas melodías, se integran a las m viejas melodías (melodías iniciales), generando un conjunto de $2m$ melodías. De este nuevo conjunto, se toman las $m/2$ mejores soluciones en términos de la función objetivo y las $m/2$ más diversas (las soluciones más alejadas de la mejor solución conocida por el compositor aislado con base en la distancia euclidiana), para formar el nuevo conjunto de m melodías que el compositor conservará. Si se repiten dos soluciones en el nuevo conjunto final de melodías, entonces una de ellas, se remplaza con una melodía (solución) generada de forma aleatoria. Además, por cada melodía producida, se valida si se puede integrar al arreglo de mejores soluciones. Obsérvese que las soluciones diversas corresponden a las soluciones más alejadas de la mejor solución conocida por el compositor aislado.

En la figura 3-4, se ilustra al compositor aislado; por un lado con sus mejores melodías, y por el otro, sus más diversas.

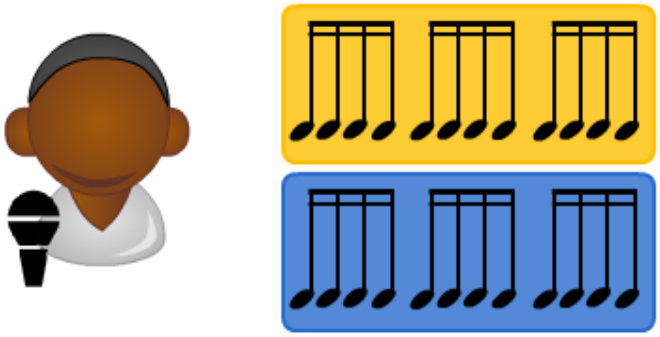

Figura 3-4: Compositor Aislado

3.2. Reactividad

3.2.1. Agentes Reactivos

Un agente reactivo es un elemento vigilante y participativo en el proceso de composición musical. Su objetivo es recopilar información de las decisiones efectuadas a lo largo del proceso de búsqueda, y con base en dicha información (memoria), tomar las mejores decisiones para generar soluciones de buena calidad, que lleven a la metaheurística a encontrar buenas (mejores) soluciones en el menor número de pasos.

Existen tres agentes reactivos inmersos en el proceso de composición musical, que emplean mecanismos de aprendizaje basados en experiencia (memoria) y toma de decisiones en línea, auto-adaptándose (otorgando mayor prioridad a buenos resultados) según las circunstancias en un momento dado, lo que permite a la metaheurística sacar provecho de esa inteligencia emergente. Los agentes se enfocan en aspectos donde de no ser empleados de forma reactiva, tendrían que calibrarse previamente a la ejecución de la metaheurística. El control de parámetros mediante reactividad ahorra tiempo en la calibración de los mismos, considerando que el problema de calibración de parámetros es por si solo un problema NP-Duro.

Las siguientes, son las áreas en las que están involucrados los agentes reactivos:

- Reactividad en la elección de la metaheurística generadora de soluciones.

- Reactividad en la calibración de parámetros mediante HS.
- Reactividad en la dirección de movimiento de las variables.

En la figura 3-5, se ilustran los tres agentes reactivos involucrados en el proceso de composición músical. El primero (a la izquierda), debe decidir entre las cinco metaheurísticas generadoras de soluciones disponibles (íconos a su izquierda) como mecanismo para generar nuevas melodías. El segundo (al centro), crea configuraciones de parámetros mediante HS para la metaheurística seleccionada por el primer agente. Y finalmente, el tercero (a la derecha), decide la dirección (izquierda o derecha) de los movimientos de las variables en situaciones de perturbación. Posteriormente, se describirá cada uno a detalle.

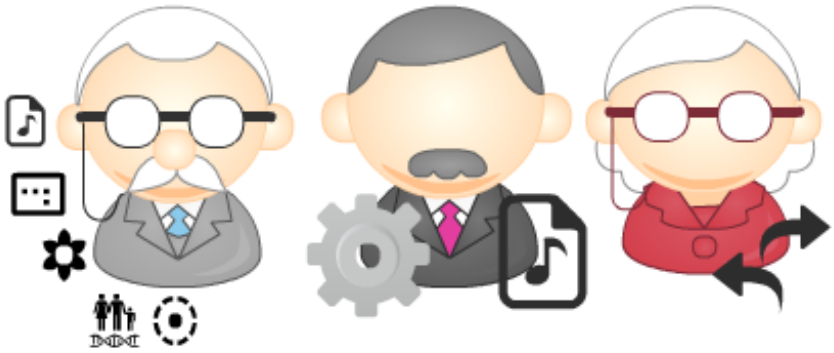

Figura 3-5: Agentes Reactivos

3.2.2. Reactividad en la elección de la metaheurística generadora de soluciones

Sin importar el tipo de comportamiento del compositor (colaborativo, parásito o aislado), cuando empieza el proceso de composición, el agente reactivo tiene la tarea de elegir un método para generar nuevas soluciones. El método que elija influye en la calidad de sus melodías; por esta razón debe elegir la técnica a emplear con base en los resultados obtenidos en sus experiencias previas. Las técnicas propuestas en este contexto se inspiran en metaheurísticas del estado del arte que emplean sus propias peculiaridades (elección de vecindarios, mecanismos de cruza, perturbaciones, etc.) para generar hermosas melodías (buenas soluciones que reduzcan o incrementen el valor de la F.O). Las cinco metaheurísticas generadoras de soluciones empleadas se eligieron con base en los buenos resultados obtenidos sobre instancias del problema de optimización global.

- Colonia de Abejas Artificiales(ABC)
- Búsqueda Armónica (HS)
- Búsqueda Dispersa (SS)
- Evolución Diferencial (ED)
- Algoritmo Genético (AG)

En la figura 3-6, se ilustra al agente reactivo encargado de la elección de la metaheurística generadora de soluciones.

Figura 3-6: Elección de la metaheurística generadora de soluciones

3.2.3. Reactividad en la calibración de parámetros mediante Búsqueda Armónica

Por otro lado, la calibración de parámetros mediante HS tiene como objetivo generar las mejores configuraciones de parámetros de entrada de cada una de las metaheurísticas generadoras de soluciones. Cada una de ellas tiene su propia memoria, con la cual, el agente reactivo crea nuevas configuraciones basándose en los principios de HS:

- Aceptación de memoria armónica.
- Ajuste de tono en un ancho de banda.
- Aleatoriedad.

Al incluir en la memoria las mejores configuraciones, después de un cierto número de iteraciones empleando HS, se tendrán configuraciónes lo suficientemente buenas como para que cada metaheurística pueda producir soluciones de alta calidad. Es importante mencionar que una solución de HS representa una configuración de parámetros asociada a una metaheurística generadora de soluciones, por lo cual será llamada "solución-configuración". El número de variables, así como el tipo de dato de cada variable (real, entero, booleano etc.) de una solución-configuración dependen de los parámetros de entrada de cada metaheurística generadora de soluciones.

En la figura 3-7, se ilustra al agente reactivo encargado de producir soluciones-configuraciones mediante HS.

Figura 3-7: Calibración de parámetros mediante Búsqueda Armónica

3.2.4. Reactividad en la dirección de movimiento de las variables

HS en su forma canónica incluye la alteración de una nota (variable) en un rango o ancho de banda acotado. Esta modificación aleatoria y local de algunas notas es un proceso similar a la

mutación en AG. Los movimientos se pueden ver como perturbaciones aportadas a una solución con la finalidad de realizar una búsqueda en su entorno. En la metaheurística Composición en una Sociedad de Músicos, al momento de producirse un movimiento perturbador, se registra la dirección del movimiento de la variable ya sea positivo (derecha) o negativo (izquierda). Se define previamente un ancho de banda en el cual se mantiene la perturbación. Por ejemplo, si el valor actual de la variable es 5 y el ancho de banda se define en +1 y -1, entonces los límites inferior y superior aceptables del movimiento son 4 y 6. Dos posibles valores resultantes de la perturbación son por ejemplo: 5.6 (positivo) o 4.56 (negativo). El agente reactivo registra dichos movimientos para aprender de los aciertos y errores y, en un futuro, tomar mejores decisiones en cuanto a la dirección hacia donde se han encontrado las mejores soluciones. Es importante mencionar que el aprendizaje es individual por cada variable (nota).

En la figura 3-11, se ilustra al agente reactivo que vigila, registra y aprende de los movimientos por cada variable, en caso de movimientos perturbantes.

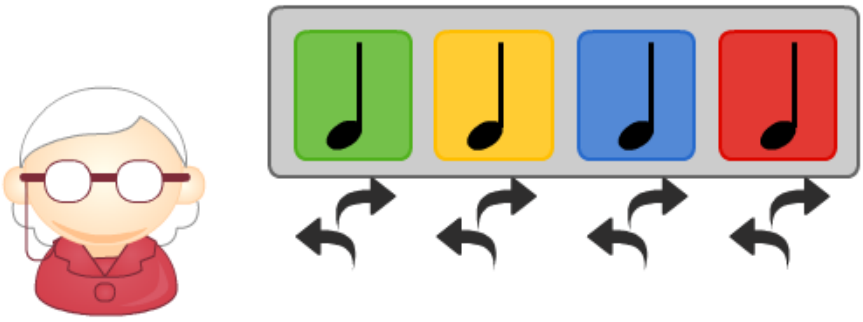

Figura 3-8: Dirección de movimiento de las variables

3.2.5. Inicialización de los agentes reactivos

Memoria de metaheurísticas generadoras de soluciones: Se crea un vector de ponderación de números enteros. El tamaño depende del número de metaheurísticas generadoras de soluciones que se utilicen en el proceso de composición musical. El valor inicial de cada casilla es 1, que corresponde a la calificación otorgada a cada metaheurística generadora de soluciones. Cada vez que una técnica produce una "buena solución", se califica positivamente sumándole 1 a la casilla del vector de ponderación correspondiente. De igual manera, en caso de que la solución sea "mala", se le califica negativamente restándole 1. Siempre se cuida que el menor valor asignado por casilla dentro del vector de ponderación sea 1. La calidad de una melodía (buena o mala) depende del criterio de aceptación de cada tipo de compositor (colaborativo, parásito o aislado). La metaheurística con un mayor puntaje tiene por lo tanto mayores probabilidades de ser elegida como técnica para creación de nuevas melodías, con base en un mecanismo de ruleta.

En la figura 3-9, se ilustra la memoria de metaheurísticas generadoras de soluciones, recién creada.

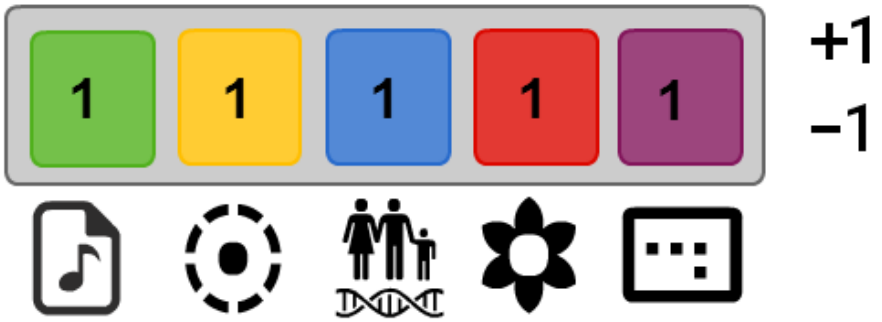

Figura 3-9: Memoria de metaheurísticas generadoras de soluciones

Memoria de soluciones-configuraciones: Una solución-configuración (solución de HS que representa una configuración de parámetros para una metaheurística generadora de soluciones) se representa por $x = (x_1, x_2, x_3, \ldots, x_d)$, donde d es la dimensión o número de parámetros a ajustar en la metaheurística generadora. La memoria se compone de m soluciones-configuraciones (donde m es un parámetro de entrada de CSM). Cada metaheurística generadora de soluciones tiene su propia memoria adaptada al número y tipo de parámetros de entrada que requiere. Por ejemplo, AG tiene 4 parámetros de configuración que son: tamaño de la población (tipo entero entre 2 e ∞), tasa de mutación (real entre 0 y 1), tasa de supervivencia (real entre 0 y 1) y rango de mutación o ancho de banda (real entre 0 e ∞); mientras que ABC tiene solo dos que son: tamaño de la población (tipo entero entre 2 e ∞) y rango de búsqueda o ancho de banda (real entre 0 e ∞). Por lo tanto, cada memoria es diferente dependiendo de la metaheurística considerada; a su vez, cada tipo de parámetro tiene un espacio de movimiento definido (límite superior e inferior) y un rango de ajuste (cuando HS emplea ajuste de tono). La calificación que se le asigne a una solución-configuración dependerá del desempeño de la misma; entendiendo como desempeño, un análisis de la calidad de las melodías (soluciones) producidas con la misma solución-configuración. A continuación, se describen los pasos a seguir para crear la memoria de soluciones-configuraciones:

- Por cada metaheurística generadora de soluciones se crea un arreglo de m soluciones-configuraciones donde m es un parámetro de configuración del CSM que indica también el número de melodías por compositor.

- Cada solución-configuración es generada de forma aleatoria y se le asigna una calificación inicial muy mala, es decir si es un problema de maximización un valor extremadamente pequeño y si es un problema de minimización un valor extremadamente grande. Los valores que se le asignen a cada nota (variable) de cada solución-configuración dependen del tipo de dato y rango permitido de variación (límite superior e inferior) de los parámetros de configuración de cada metaheurística generadora de soluciones.

En la figura 3-10, se ilustra la memoria de soluciones-configuraciones. Cada metaheurística generadora de soluciones tiene su propia memoria en donde se producen las soluciones-configuraciones que servirán de parámetros de entrada.

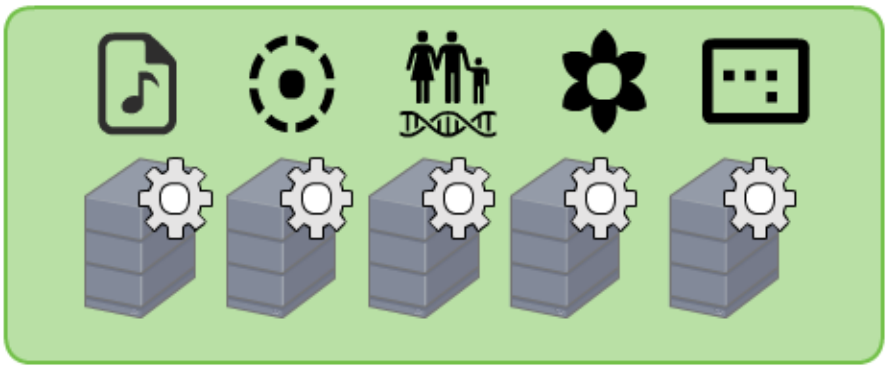

Figura 3-10: Memoria de soluciones-configuraciones

Memoria de movimiento de variables: Cuando se produce un movimiento perturbatorio durante la creación de una melodía (solución), se registra la dirección de movimiento por cada variable del vector solución. Es decir, se mantiene un registro de los movimientos a la izquierda y derecha que fueron éxito y fracaso por cada nota (variable) de la melodía (solución) cuando suceda un movimiento perturbante. El fracaso o el éxito depende de la calidad de la solución, es decir, si el compositor la acepta como suya dentro de su memoria o no. Por ejemplo: la nueva melodía es aceptada por el compositor y consta de dos variables; además, se produce un movimiento perturbante hacia la izquierda en la segunda variable, entonces, se añade un punto a la calificación del movimiento a la izquierda de la segunda variable. En caso de fracaso se resta un punto.

La figura 3-11, ilustra la memoria de movimiento de variables. El arreglo superior registra los movimientos hacia la derecha y el inferior a la izquierda, por cada variable del vector solución (melodía).

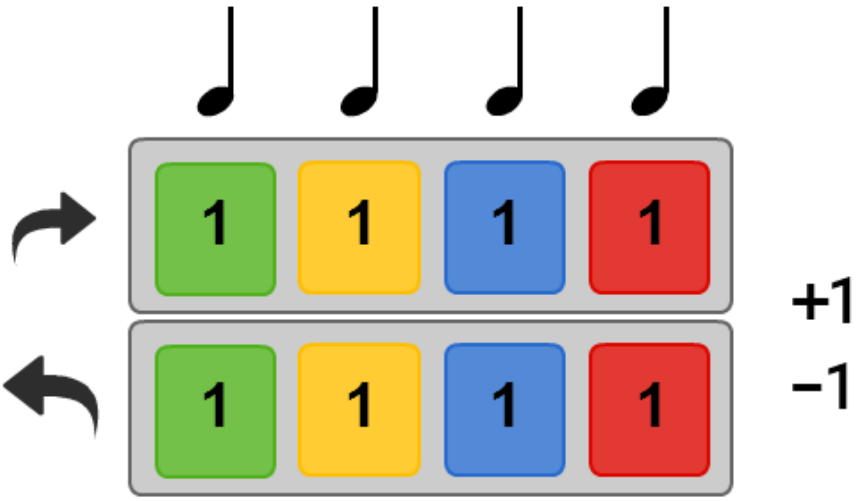

Figura 3-11: Memoria de movimiento de variables

3.2.6. Algoritmo del proceso reactivo musical

A continuación, se presenta el procedimiento empleado por un compositor (colaborativo, aislado o parásito) para crear un conjunto de nuevas melodías, que en principio se espera que sean mejor que la peor melodía del compositor considerado. Lo ideal es que se integren al arreglo de mejores melodías y en el mejor de los casos, que se genere la mejor melodía. Este

proceso contempla premiar a las técnicas que producen buenas soluciones y castigar a las otras; el registro de dicha acción persiste en las memorias de los agentes reactivos.

Algoritmo 3 Algoritmo del proceso reactivo musical

```
1: Selección de metaheurística generadora de soluciones
2: Creación de solución-configuración mediante Búsqueda Armónica [Ver algoritmo]
3: para cada melodía a componer (dependiendo el tipo de compositor) hacer
4:    Crear melodía
5:    Si al crear la nueva melodía sucede un movimiento perturbante se registra
6:    Calificar la nueva melodía
7:    si la melodía fue calificada como "buena" entonces
8:       Aplicar técnica de reemplazo del compositor
9:       Premiar metaheurística
10:      Premiar movimientos perturbantes
11:   si no
12:      Se desecha la melodía
13:      Castigar metaheurística
14:      Castigar movimientos perturbantes
15:   fin si
16: fin para
17: Calificar solución-configuración
```

A continuación, se describen cada uno de lo pasos del algoritmo:

- *Selección de metaheurística generadora de soluciones:* Se efectúa con base en la memoria de metaheurísticas generadoras de soluciones mediante el método de la ruleta, donde la metaheurística con mayor puntaje tendrá mayor probabilidad de ser elegida.

- *Crear melodía:* Una vez que:

  - se ha elegido la metaheurística generadora de soluciones
  - se tiene la solución-configuración producida por HS
  - se tienen las melodías base (melodías del compositor)

  La metaheurística utiliza sus propios procedimientos de búsqueda para generar la nueva melodía.

- *Calificar melodía:* se califica con base en los criterios de aceptación del compositor considerado.

- *Premiar metaheurística:* Se premia a la metaheurística generadora de soluciones sumando un punto a la casilla correspondiente en el vector de ponderaciones.

- *Premiar movimientos perturbantes:* Si ocurrió un movimiento perturbante en una o más variables, se premia el movimiento (izquierda o derecha) para dicha(s) variable(s) sumando un punto en su(s) casilla(s) correspondiente(s).

- *Castigar metaheurística:* Se castiga a la metaheurística generadora de soluciones restándole un punto a la casilla correspondiente en el vector de ponderaciones. Se cuida que el menor valor sea 1.

- *Castigar movimientos perturbantes:* Si ocurrió un movimiento perturbante en una o más variables, se castiga el movimiento (izquierda o derecha) para dicha(s) variable(s) restando un punto en su(s) casilla(s) correspondiente(s).
- *Calificar solución-configuración:* La calificación que se le asigne a la solución-configuración, depende de la calidad de las melodías generadas, y es el promedio del número de llamadas la función objetivo que se necesitaron para generar la melodía (solución). Si la calificación es mejor en comparación con la peor solución-configuración en la memoria de soluciones-configuraciones correspondiente a la metaheurística generadora de soluciones seleccionada, entonces se reemplaza.

La figura 3-12, ilustra el proceso reactivo musical empleado por el compositor (ya sea colaborativo, aislado o parásito) encargado de componer nuevas melodías. Se indica en cada paso el número correspondiente con el número de línea en el algoritmo del proceso reactivo musical, que se explicó anteriormente.

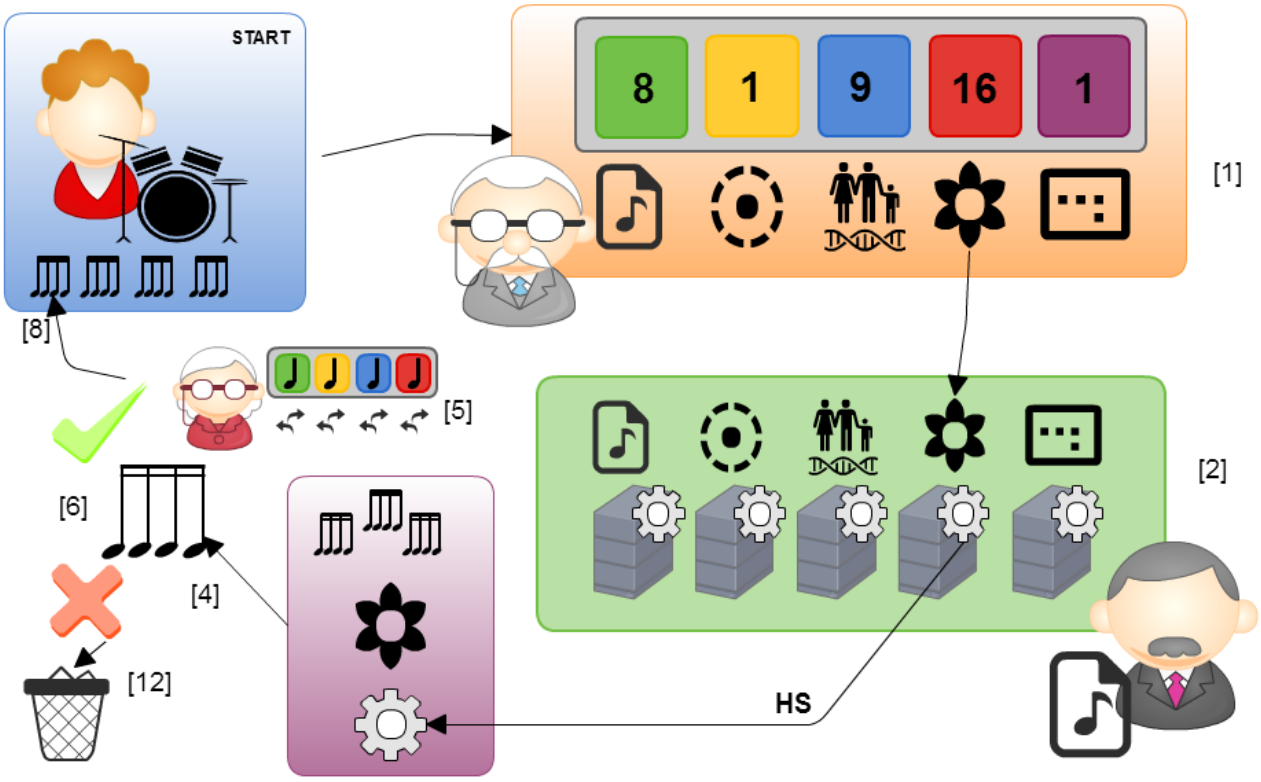

Figura 3-12: Proceso reactivo musical

3.2.7. Creación de solución-configuración mediante Búsqueda Armónica

Una solución-configuración es la configuración de parámetros de entrada que la metaheurística generadora de soluciones ocupa para generar una melodía. Cada metaheurística generadora tiene su propia memoria de soluciones-configuraciones. Este procedimiento se inspira en la metaheurística de HS para crear una solución-configuración, al contemplar tres opciones para asignar un valor a cada nota (variable) i del nuevo vector solución de tamaño d ($1 \leq i \leq d$). A continuación, se describen estas tres opciones:

1. *Aceptación de la memoria armónica:* Recordar características pasadas. Se elige de manera aleatoria una solución-configuración de la memoria armónica y se toma el valor de la

variable i para asignarlo en el valor de la variable i de la nueva solución-configuración aspirante.

La figura 3-13, ilustra cómo la primera casilla del vector solución es asignada recordando características pasadas, en este caso el color rojo, que ya estaba contenido en la memoria armónica.

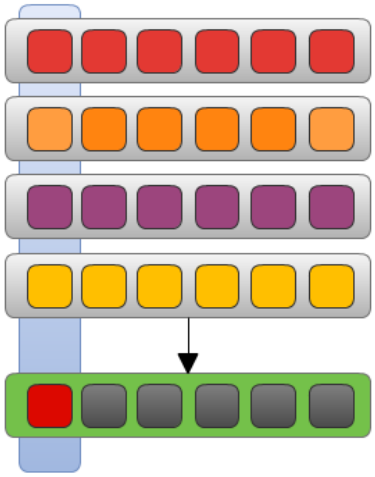

Figura 3-13: Recordar características pasadas

2. *Creatividad:* Generar características de manera aleatoria. Se genera un valor aleatorio entre el límite inferior y el límite superior del espacio de variación del parámetro y se asigna el valor a la variable i de la nueva solución-configuración aspirante.

   La figura 3-14, ilustra como la segunda casilla del vector solución es asignada siendo creativo; es decir, se produce una característica nueva, que en este caso es el color azul, el cual no estaba contenido en la memoria armónica previamente.

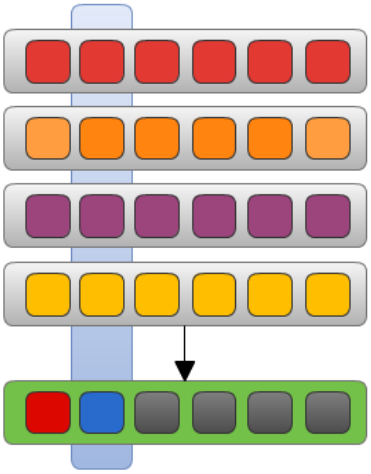

Figura 3-14: Generar características de manera aleatoria

3. *Ajuste de tono en ancho de banda:* Recordar características pasadas, alterando ligeramente. Se elige de manera aleatoria una solución-configuración de la memoria armónica, se toma el valor de la variable i y se ajusta el valor de manera aleatoria dentro del ancho de banda. Finalmente, se asigna al valor a la variable i de la nueva solución-configuración aspirante.

La figura 3-15, ilustra como la tercer casilla del vector solución es asignada recordando una característica pasada, pero alterando un poco. Se recuerda el color purpura, pero se modifica un poco y el resultado es el color lila (muy parecido al purpura).

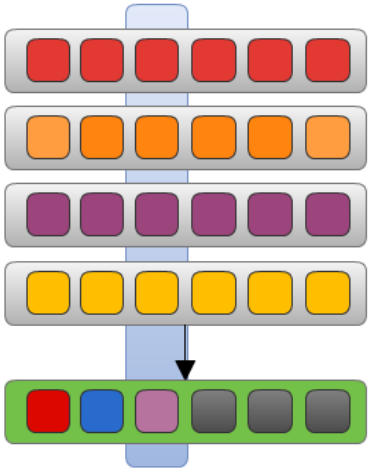

Figura 3-15: Recordar características pasadas, alterando ligeramente

A continuación, se presenta el algoritmo para crear una solución-configuración mediante HS:

Algoritmo 4 Creación de solución-configuración mediante HS

1: Se crea un vector solución vacío de tamaño d denotado como "solución-configuración aspirante"
2: **para** cada variable i del vector solución (desde *1* hasta d) **hacer**
3: **si** aleatorio1 $>$ $r_{aceptacion}$ **entonces**
4: Aceptación de la memoria armónica
5: **si no**
6: **si** aleatorio2 $>$ r_{ajuste} **entonces**
7: Ajuste de tono en ancho de banda
8: **si no**
9: Creatividad
10: **fin si**
11: **fin si**
12: **fin para**
13: La solución-configuración aspirante está lista para ser introducida como parámetros de entrada en una metaheurística generadora de soluciones.

Donde:

- $r_{aceptacion}$: Parámetro fijo que determina la probabilidad de que se utilicen elementos de la memoria armónica. En caso contrario, se realiza un ajuste de tono o se generan características de forma aleatoria. El valor utilizado en el marco de este trabajo es $r_{aceptacion} = 0.3$.

- r_{ajuste}: Parámetro fijo que determina la probabilidad de que se realice un ajuste de tono. En caso contrario, se generan características de forma aleatoria. El valor utilizado en el marco de este trabajo es $r_{ajuste} = 0.1$.

- aleatorio1 y aleatorio2: Son números reales e independientes, generados de forma aleatoria con una distribución uniforme entre 0 y 1.

3.3. Metaheurísticas generadoras de soluciones

A continuación, se presenta el algoritmo adaptado de cada una de las metaheurísticas empleadas para generar nuevas soluciones dentro del proceso reactivo musical, es decir, las llamadas "metaheurísticas generadoras de soluciones". Recuérdese que dicha adaptación se debe a que las metaheurísticas seleccionadas generan en cada iteración más de una solución (a excepción de HS); en cambio, CSM necesita sóla una solución a la vez.

3.3.1. Metaheurística generadora de soluciones basada en ABC

A continuación, se presenta el algorimo adaptado de la metaheurística generadora de soluciones basada en ABC:

Algoritmo 5 ABC Adaptado

Entrada:

0: rb: Rango de Búsqueda local (qué tanto puede volar la abeja en una búsqueda local).
0: a: Solución de partida, en la cual se posiciona la abeja supervisora y su obrera inicialmente.
0: z: Mejor solución conocida o mejor fuente de alimento.
1: $bl = BúsquedaLocal\ (a, rb)$, $bl \geq a$
2: **si** $bl < z$ **entonces**
3: $bd = BúsquedaDirigida\ (bl)$, $bd \geq bl \geq a$
4: **si** $bd == a$ **entonces**
5: $be = SoluciónAleatoria()$
6: **devolver** be
7: **si no**
8: **si** $bd > z$ **entonces**
9: $z \leftarrow bd$
10: **devolver** bd
11: **si no**
12: **devolver** bd
13: **fin si**
14: **fin si**
15: **si no**
16: $z \leftarrow bl$
17: **devolver** bl
18: **fin si**

A continuación, se describen las tres estrategias del proceso de búsqueda:

1. Búsqueda Local: Inicialmente la abeja supervisora manda a su obrera a buscar alimento en una zona acotada; el objetivo es encontrar una mejor fuente de alimento que la mejor que se conoce al inicio.

2. Búsqueda Dirigida: Si la abeja obrera no encuentra algo mejor que la mejor fuente de alimento conocida en la búsqueda local, la supervisora le ordena realizar una búsqueda dirigiéndose hacía la mejor fuente de alimento conocida, en ese recorrido puede ser que encuentre una mejor fuente de alimento. Aunque la obrera no encuentre algo mejor que la mejor fuente de alimento conocida, se tomará como una búsqueda exitosa haber encontrado algo mejor que con lo que se tenía antes de realizar la búsqueda local.

3. Búsqueda Aleatoria: Cuando la búsqueda local y la búsqueda dirigida no consigan mejorar en nada la solución de partida, la abeja supervisora se convierte en abeja exploradora y

se dirige a otra nueva posición de manera aleatoria, llevándose consigo a su obrera para iniciar su búsqueda desde otro punto, ya que el anterior no fue de utilidad.

En la figura 3-16, se muestran las tres fases de búsqueda en ABC:

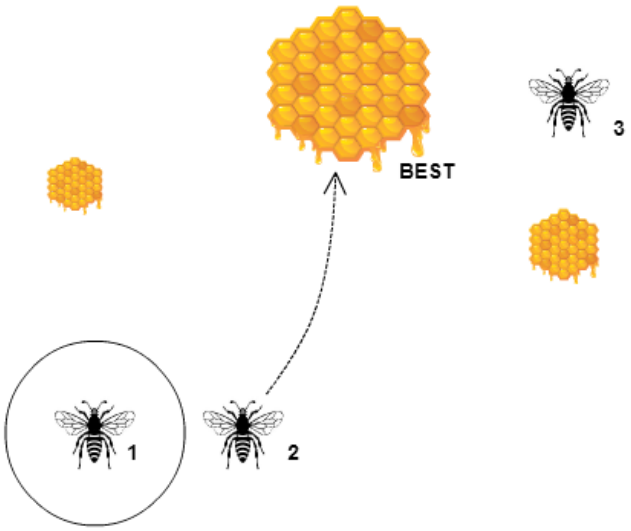

Figura 3-16: Fases del proceso de búsqueda en ABC

3.3.2. Metaheurística generadora de soluciones basada en AG

A continuación, se presenta el algoritmo adaptado para la metaheurística generadora de soluciones basada en AG, en donde se introduce una segunda alternativa de búsqueda (Búsqueda local), en caso de que la mutación no produzca un mejor individuo:

Algoritmo 6 AG Adaptado

Entrada:

0: *tasaMutacion* Probabilidad de que ocurra una mutación.
0: *rangoMutacion* Que tanto podrán cambiar las variables si ocurre una mutación.
1: Elegir al *padre1* mediante torneo binario.
2: Elegir al *padre2* mediante torneo binario.
3: **si** *padre1* == *padre2* **entonces**
4: *hijo* = *padre1*
5: **si no**
6: *hijo* = *Cruza* (*padre1* , *padre2*)
7: **fin si**
8: *clon* = *Clonar* (*hijo*)
9: *mutante* = *Mutar* (*clon*, *tasaMutacion* , *rangoMutacion*)
10: **si** *mutante* es mejor que *hijo* **entonces**
11: **devolver** *mutante*
12: **si no**
13: *hijo* = *BúsquedaLocal* (*hijo*, *rangoMutacion*)
14: **devolver** *hijo*
15: **fin si**

La figura 3-17, ilustra la selección de los individuos padre (torneo binario); posteriormente, la cruza que genera un hijo, el cual es clonado y posteriormente mutado. En caso de que el mutante no sea mejor que el hijo, se realiza una búsqueda local a partir del hijo.

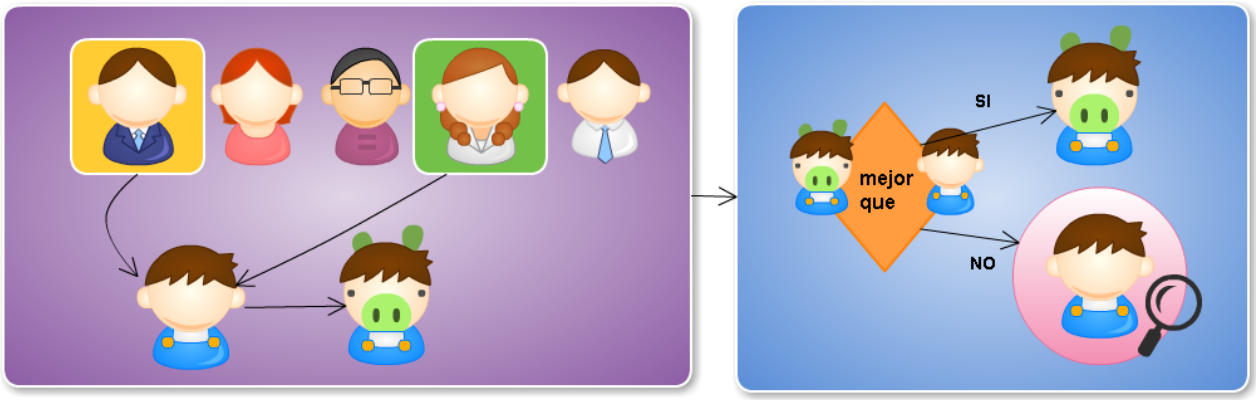

Figura 3-17: AG Adaptado

3.3.3. Metaheurística generadora de soluciones basada en ED

La adaptación del algoritmo de ED descarta la etapa de "Inicialización", ya que partimos de una solución a del dominio del compositor y también se descarta la etapa de "Selección", ya que la nueva solución generada será evaluada con base en los propios criterios del compositor.

A continuación, se presenta el algoritmo adaptado para la metaheurística generadora de soluciones basada en ED:

Algoritmo 7 ED Adaptado

Entrada:

0: a Solución de partida
0: CR Factor de Cruza [0,1]
0: $parCtrl$ Parámetro de control [0,1]
1: Seleccionar dos melodías de la experiencia del compositor (b y c) tal que $b \neq c, a \neq b, a \neq c$, es decir que sean elementos distintos entre si;
2: $mutante = a + (Pctrl)(b - c)$
3: $recombinado = CruzaDiferencial\ (a,\ mutante,\ CR)$
4: **devolver** *recombinado*

Donde:

- Mutación: Sea a la solución de partida o "Individuo original", y P la población (melodías de la experiencia del compositor), donde $a \in P$. Se debe de elegir dos elementos de P : b, c tal que $b \neq c, a \neq b, a \neq c$, es decir que sean elementos distintos entre si; se debe de cumplir que $|P| \geq 3$. Se genera una nueva solución x, realizando la siguiente operación, tomando a las soluciones a, b, c como vectores: $x = a + (Pctrl)(b - c)$, donde $Pctrl$ es un valor real entre 0 y 1. La nueva solución x, se conoce como "Mutante".

- Si al momento de realizar la mutación, los valores sobrepasan los límites del espacio de búsqueda, se resta o se suma en dirección contraria al límite sobrepasado hasta que el valor este dentro del rango.

- Recombinación: Se hace uso de la técnica "Cruza diferencial" en donde se combinan elementos (variables) del "Individuo original" con su "Mutante" con base en el factor de cruza que determina qué tantas características mutantes se le otorgan al "Nuevo individuo".

A continuación, se presenta el algoritmo de Cruza Diferencial:

Algoritmo 8 Algoritmo de Cruza Diferencial

Entrada:

0: a Solución inicial
0: *mutante* Solución mutante CR Factor de Cruza [0,1]
1: pos = Número entero aleatorio entre 1 y D, donde D es la dimensión del vector solución o número de variables de la solución.
2: $posPartida = pos$
3: **repetir**
4: $a[pos] = mutante[pos]$
5: **si** $pos == D$ **entonces**
6: $pos = 1$
7: **si no**
8: $pos = pos + 1$
9: **fin si**
10: **si** $posPartida == pos$ **entonces**
11: **devolver** a
12: **fin si**
aleatorio = aleatorio[0,1]. Función generadora de aleatorios entre 0 y 1
13: **hasta que** aleatorio menor que CR
14: **devolver** a

La figura 3-18, ilustra la fase de mutación y recombinación del algoritmo de ED Adaptado. En el primer recuadro se observa cómo se llava acabo la mutación con tres individuos distintos mediante operaciones vectoriales. En el segundo recuadro, el individuo final es el resultado de la recombinación mediante cruza diferencial del mutante y el individuo original a.

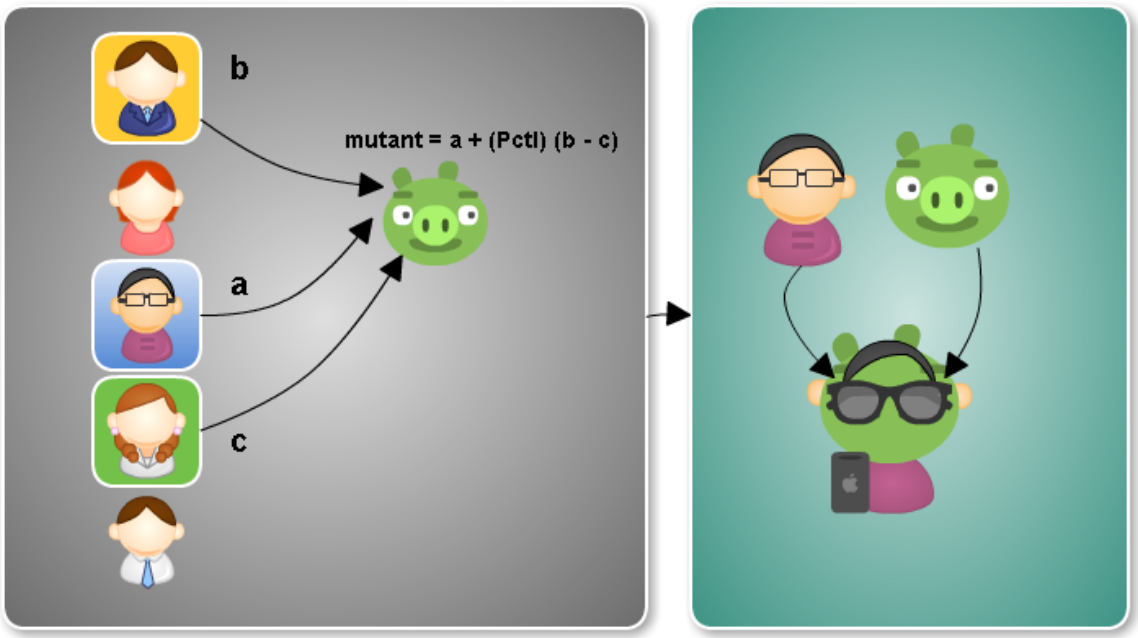

Figura 3-18: Mutación y Recombinación

3.3.4. Metaheurística generadora de soluciones basada en SS

A continuación, se presenta el algoritmo adaptado para la metaheurística generadora de soluciones basada en SS:

Algoritmo 9 SS Adaptado

Entrada:

0: *CR* Tamaño del Conjunto de Referencia
0: *rb* Rango de Búsqueda (Búsqueda local)
1: *MEJORES[]* = Seleccionar los *CR* /2 mejores elementos (melodías) de la experiencia del compositor.
2: *DIVERSOS[]* = Seleccionar los *CR* /2 elementos de la experiencia del compositor más distantes de la mejor solución.
3: *MEJOR* = Elegir aleatoriamente un elemento de entre los *MEJORES[]*
4: *DIVERSO* = Elegir aleatoriamente un elemento de entre los *DIVERSOS[]*
5: *Solución* = *Cruzar* (*MEJOR*, *DIVERSO*)
6: *Solución* = *BúsquedaLocal* (*Solución*, *rb*)
7: **devolver** *Solución*

Donde:

1 La mitad de los CR individuos está conformada por las mejores melodías de la experiencia del compositor, es decir, los que tienen mejor evaluación con respecto a la función de costo.

2 La otra mitad se constituye de los elementos más diversos, se seleccionan los más alejados de la mejor solución con base a alguna métrica, en este trabajo se empleó la métrica euclidiana. Se entiende como soluciones diversas, aquellas que están más alejadas de la mejor solución. En este escenario pueden ocurrir dos situaciones: que la solución diversa sea también una buena solución o que no sea de calidad, pero lo que realmente se busca es que esté lo suficientemente alejada como para trazar una ruta desde los dos puntos y en ese camino encontrar mejores soluciones. Si un individuo se repite tanto en los mejores como en los diversos se elimina el duplicado y se reemplaza por una solución aleatoria.

3,4 Se elige de manera aleatoria un elemento "buenoz un elemento "diverso"del conjunto de referencia.

5 Cruzar ambos elementos (bueno y diverso), de la cruza resulta un nuevo elemento.

6 A partir del nuevo elemento producido se realiza una búsqueda local para intensificar la búsqueda.

La figura 3-19, ilustra los pasos del algoritmo SS Adaptado. Inicialmente se cuenta con un conjunto grande de soluciones. Posteriormente se forma el subconjunto *CR* conformado por las mejores (melodías amarillas) y más diversas (melodías azules) soluciones. Se selecciona una melodía de cada tipo, se realiza la cruza y al final se realiza una búsqueda local.

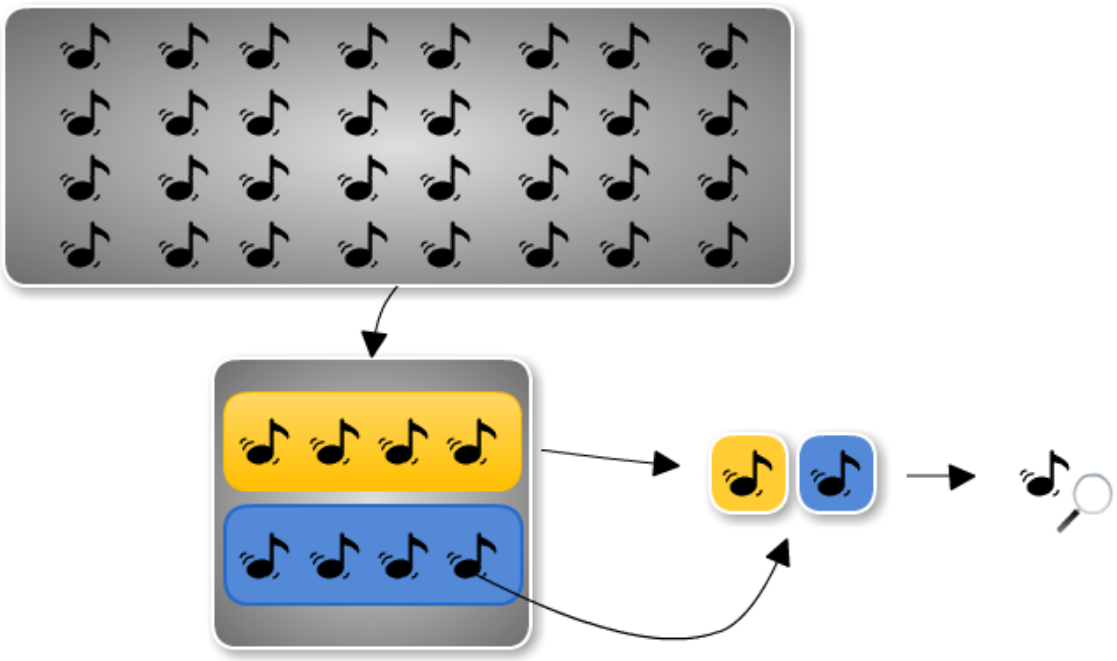

Figura 3-19: SS Adaptado

Capítulo 4

Calibración de parámetros

4.1. Introducción

La calibración de parámetros es un problema de optimización en el cual se desea encontrar la mejor combinación de parámetros de entrada para una metaheurística, con la cual, la metaheurística encuentre una solución de buena calidad en el menor tiempo posible y de forma constante.

Debido a que los parámetros de entrada de una metaheurística tienen una gran influencia en su efectividad (alcanzar el óptimo) y eficiencia (tiempo), la calibración de parámetros tiene una gran importancia al momento de probar y ejecutar las metaheurísticas desarrolladas; de este procedimiento depende el tiempo de ejecución que le tome a la metaheurística encontrar el óptimo (si lo encuentra), así como la eficiencia para direccionar la búsqueda a las mejores soluciones.

La necesidad de calibrar múltiples parámetros, combinado con la naturaleza estocástica de la metaheurística, hacen de la calibración de parámetros un problema no trivial.

4.2. Calibración de parámetros

Aunque la calibración de parámetros introduce mayor flexibilidad y robustez en la resolución de problemas de optimización, requiere un cuidadoso procedimiento al momento de realizarse. Un error común es creer que encontrando una calibración adecuada para un problema, esta misma configuración funcionará para otros problemas. Cada problema o instancia tiene sus propias características y paisajes de búsqueda diferentes, es por esta razón, que se debe de calibrar parámetros para cada nuevo problema a resolver, aunque la metaheurística sea la misma.

Se ha demostrado que la calibración de parámetros es un problema NP-Duro por sí mismo, y es claro, ya que es un problema combinatorio en donde se busca encontrar la combinación de valores asignados a los parámetros de la metaheurística, que minimice las llamadas a la función objetivo y que encuentre una buena solución o la mejor en un tiempo razonable.

Existen dos estrategias diferentes para la calibración de parámetros.

- *Calibración de parámetros fuera de línea (meta-optimización)*:
    - Los valores de los diferentes parámetros se establecen antes de la ejecución de la metaheurística.
    - El programador debe efectuar un trabajo previo para encontrar la mejor configuración que minimice las llamadas a la función objetivo y por consiguiente mejorar en tiempos.

- *Estrategias en línea de calibración de parámetros (control de parámetros)*:

  - Se controlan los parámetros y se actualizan de forma dinámica o adaptativa durante la ejecución de la metaheurística.
  - El programador se olvida de la calibración de parámetros ya que el propio algoritmo trata de encontrar la mejor configuración durante la ejecución.

A continuación, se mencionan dos mecanismos para la calibración de parámetros fuera de línea.

1. Métodos estadísticos

Basados en el diseño de experimentos, con el cual se obtiene la mayor cantidad de información posible con el menor número de experimentos, también llamados métodos exactos.

Por cada parámetro a calibrar se establece un vector de valores de prueba. Se prueban todas las combinaciones posibles entre ellos, calculando el promedio o la media de las llamadas a la función objetivo o el tiempo de ejecución de n corridas con la misma calibración para establecer la eficiencia de la combinación. Algunas técnicas basadas en el diseño de experimentos involucran mecanismos de corte (fraccionales), para no experimentar en combinaciones que no tienen posibilidad de arrojar buenos resultados, y de esta forma reducir el número de experimentos.

2. Optimización

Encuentran la mejor combinación mediante el uso de una metaheurística. Como se ha mencionado anteriormente, la calibración de parámetros es un problema NP-Duro. Una de las ventajas de las metaheurísticas sobre los métodos exactos, es que se pueden aproximar o encontrar la mejor solución en un tiempo razonable. De esta forma se automatiza la calibración y se le deja el trabajo a la metaheurística, lo cual resulta ser conveniente por cuestiones de tiempo y esfuerzo.

Cualquier metaheurística del estado del arte puede ser adaptada para realizar la función de calibrador de parámetros, ya que saben guiar la búsqueda para encontrar soluciones de buena calidad o, en el mejor de los casos, encontrar la mejor.

4.3. Búsqueda Armónica como mecanismo de calibración de parámetros fuera de línea

4.3.1. Introducción

Una metaheurística es un método inteligente que por si misma, ya que es capaz de encontrar buenas soluciones a un problema de optimización implementando estrategías de búsqueda. En este sentido, dejar que una metaheurística, se encargue de la calibración de parámetros sin supervisión, resulta una propuesta atractiva.

Una técnica concurrida al momento de calibrar los parámetros de una metaheurística (fuera de línea) es variar un parámetro dentro de un rango de valores y dejar otros estáticos. La mejor combinación es aquella donde se obtiene el mejor desempeño de la metaheurística, una vez que se probaron todos los valores posibles del parámetro que se puso a variar. Este proceso es un estira y encoge, ya que al momento de variar un parámetro y dejar otro fijo, al paso del tiempo se obtendrá una combinación que sea la mejor, pero posteriormente al mover otro parámetro que se había quedado fijo, se puede obtener un mejor resultado, y así sucesivamente. Es claro

que este mecanismo no es la mejor forma de calibrar los parámetros de una metaheurística, pero se puede llegar a una configuración decente.

Pero ahora, imagínese que se guardan un conjunto de buenas configuraciones y con base en estas se calibran los parámetros de una metaheurística. Naturalmente, con el paso del tiempo estas configuraciones comenzarón a dirigirse hacia la mejor configuración y parecerse entre si. Para evitar estancamientos en óptimos locales se puede diversificar con un ajuste (movimiento moderado) o un toque de aleatoriedad para buscar en otras direcciones que no se habían explorado. De esta forma se guia la búsqueda (calibración) de una forma inteligente, en comparación al proceso de estira y encoge descrito en un principio.

Como se puede observar, este procedimiento se asemeja a una técnica de búsqueda moviéndose en un espacio acotado con el objetivo de encontrar la mejor solución. Esta misma acción la realizan muchas de las metaheurísticas descritas en este trabajo, pero sólo una se parece en el hecho que en algunas ocasiones permanecen valores fijos, otras ocasiones se mueven en un ancho de banda y otras se eligen valores de forma aleatoria. Es claro que se habla de Búsqueda Armónica.

4.3.2. Solución-configuración

Si HS es la técnica de calibración de parámetros, una solución de HS es una nueva configuración para la metaheurística.

En este sentido se puede definir a la " solución-configuración" x como:

$x = (x_1, x_2, x_3, \ldots, x_d)$, donde d es la dimensión del vector solución, número de variables o número de parámetros de entrada de la metaheurística.

Obsérvese que las metaheurísticas tienen variables de configuración de diferente tipo de dato. Algunas variables pueden ser de tipo: entero, real, binario, carácter, etc. Por lo que a cada tipo de dato se debe delimitar en su espacio de valores que puede tomar (limite inferior y limite superior), así como la precisión en el caso las variables de tipo real. Por ejemplo, AG tiene un parámetro de tipo entero que es el tamaño de la población que puede variar entre 2 hasta n; mientras que la tasa de mutación es de tipo real y varía entre 0 y 1 con las décimas de precisión que sean convenientes. En el caso de CSM x se define como: $x = (n, m, tco, tpar)$, donde n, m, tco, $tpar$, son los parámetros de entrada de CSM.

4.3.3. Desempeño de una solución-configuración

Se ha hablado de cómo HS crea soluciones-configuraciones, pero ¿cómo saber si es una buena configuración de parámetros para CSM?. En principio se tiene que ejecutar a CSM con dicha configuración para conocer su desempeño, es decir, cuántas llamadas a la función objetivo le tomó a CSM encontrar el óptimo (considerando que se conoce el óptimo). Pero es claro que una sola ejecución no es suficiente, debido a la naturaleza estocástica de las metaheurísticas. Por ejemplo, en una ejecución se puede obtener un muy buen resultado (100 llamadas a F.O) y en la siguiente uno muy malo (10,000 llamadas a F.O) y la siguiente un resultado aceptable (500 llamadas a F.O); por lo que se debe tomar una muestra de resultados o calificaciones y tomar el promedio o la media como valor para decidir si el desempeño de la configuración fue bueno o malo. Como se ha indicado en el algoritmo, si el desempeño de la solución-configuración es mejor que la peor solución-configuración de la memoria armónica, entonces se remplaza.

Es importante mencionar que se debe de considerar lo siguiente:

- Se debe de poner un límite de llamadas a la función objetivo para cada ejecución de CSM. En algunas ocasiones la configuración puede ser tan mala que provoque que CSM nunca se alcance el óptimo, por lo que al alcanzar el límite, se detiene la ejecución del CSM,

y como el límite es un valor alto, entonces la calificación de dicha configuración es muy mala.

- Una vez que se encuentra el óptimo se detiene la ejecución del CSM y el número de llamadas a la F.O es la calificación de dicha solución-configuración.

4.3.4. Parámetros de entrada del calibrador de parámetros por HS

Si se desea calibrar los parámetros de la metaheurística que se utiliza para calibrar parámetros, se podría pensar en volver a utilizar este método, o cualquier otro, para calibrar al calibrador, pero realmente complica demasiado el proceso. Dado que HS se desempeña como calibrador de parámetros automático, se elige una configuración de parámetros adecuada que sea buena, aunque no sea la mejor.

Por esta razón, es que se fijaron los valores de los parámetros de entrada de forma estática. En este trabajo se utilizaron los valores que en la práctica se han ajustado a un mayor número de problemas resultos mediante HS, haciendo un balance entre intensificación y diversificación, de tal manera que sea posible salir de óptimos locales sin caer en la aleatoriedad y con la debida intensificación para seguir por caminos prometedores. Obsérvese que se añaden parámetros con la finalidad de no estancarse en malas configuraciones.

A continuación, se describe la configuración establecida:

- Número de pruebas por solución-configuración ($testDes$). Número de ejecuciones del CSM que se realizan por solución-configuración, con la finalidad de obtener el desempeño de la solución-configuración (Ver siguiente sección). Valor: 6.

- Máximo número de generaciones ($maxIT$): El número máximo de generaciones que el algoritmo de HS ejecutará para la calibración. Valor: 300.

- Máximo número de llamadas a la F.O ($maxFO$): El número máximo de llamadas a la F.O que CSM puede sumar en una ejecución, si el límite es alcanzado, se detiene la ejecución de CSM. Valor: Depende de la dimensión de la instancia a resolver. Para instancias con d=5 se estableció 3000, con d=10 se estableció 5000 y para d=30 se estableció 10,000.

- Tamaño de memoria armónica ($tamArm$). Número de soluciones-configuraciones que se mantienen en memoria durante la calibración. Valor: 6.

- Aceptación de memoria armónica ($r_{aceptacion}$). Probabilidad de que se recuerde una parámetro de la memoria armónica. Valor: 0.4.

- Ajuste de tono (r_{ajuste}). Probabilidad de que se recuerde un parámetro y se ajuste un poco. Valor: 0.65.

- Ancho de banda (b_{rango}): Qué tanto se altera un parámetro en caso de que se realice el ajuste de tono. Valor: Depende del tipo de dato del parámetro. Para valores enteros (n y m) es 8, considerando que el límite inferior es 3 y el superior es 80. Para valores reales (tco, $tpar$) es 0.13, considerando que el límite inferior es 0 y el superior es 1.

Además, es importante tener en cuenta las siguientes consideraciones:

- Si la tasa de aceptación de memoria armónica es muy alta, siempre se sigue el camino de buenos resultados, por lo que la búsqueda se puede quedar estancada en óptimos locales con gran facilidad.

- Si el ancho de banda es muy grande, la búsqueda se torna aleatoria, y se pierde la dirección hacia los buenos resultados.

- Si la tasa de aceptación de memoria armónica es muy baja y la tasa de ajuste de tono también, entonces la búsqueda se torna aleatoria.

- Si el tamaño de la memoria armónica es muy grande, llevará mas tiempo tener una memoria armónica confiable.

4.3.5. Algoritmo

Como ya se ha mencionado en secciones anteriores, las tres estrategias que emplea HS para dirigir la búsqueda son:

- Aceptación de memoria armónica

- Ajuste de tono en un ancho de banda

- Aleatoriedad

A continuación, se puede observar cómo estos elementos y los otros parámetros de entrada influyen en el algoritmo calibrador de metaheurísticas mediante HS:

Algoritmo 10 Calibrador HS

Entrada:

0: Número de pruebas por solución-configuración ($testDes$)
0: Máximo número de generaciones ($maxIT$)
0: Máximo número de llamadas a la F.O ($maxFO$)
0: Tamaño de memoria armónica ($tamArm$)
0: Aceptación de memoria armónica ($r_{aceptacion}$)
0: Ajuste de tono (r_{ajuste})
0: Ancho de banda (b_{rango})
1: Inicializar la memoria armónica (MA) de tamaño $tamArm$ con valores aleatorios y asignarle a cada solución-configuración calificaciones muy malas.
2: iteraciones = 0
3: **mientras** iteraciones $< maxIT$ **hacer**
4: Nueva solución-configuración = HS ($r_{aceptacion}$, r_{ajuste}, b_{rango}, MA)
5: La calificación de la nueva solución-configuración (desempeño) se obtiene ejecutando CSM $testDes$ veces y calculando el promedio de las llamadas a la F.O.
6: Si por cada ejecución de CSM se alcanzan $maxFO$ llamadas a la F.O se detiene la ejecución de CSM.
7: **si** nueva solución-configuración es mejor que la peor contenida en MA **entonces**
8: Se remplaza la peor por la nueva solución-configuración.
9: **si no**
10: Se desecha la nueva solución-configuración
11: **fin si**
12: iteraciones = iteraciones + 1
13: **fin mientras**
14: **devolver** mejor solución-configuración contenida en MA

Algoritmo 11 HS (generadora de soluciones-configuraciones)

Entrada:

0: $r_{aceptacion}$ Rango de aceptación de la memoria armónica.
0: r_{ajuste} Frecuencia de ajuste de tono.
0: b_{rango} Ancho de Banda.
0: Memoria Armónica MA
1: Se genera una solución-configuración vacía, que contiene los parámetros de entrada de CSM $x = (n, m, tco, tpar)$
2: **para** cada elemento i de x , con i desde 1 hasta 4 **hacer**
3: **si** aleatorio1 $< r_{aceptacion}$ **entonces**
4: Aceptación de la memoria armónica (Recordar). Se elige de manera aleatoria una solución-configuración de MA y se toma el valor de la variable i para asignarlo en el valor de la variable i de la nueva solución-configuración.
5: **si no**
6: **si** aleatorio2 $< r_{ajuste}$ **entonces**
7: Ajuste de tono en ancho de banda (Generar algo ligeramente diferente). Se elige de manera aleatoria una solución-configuración de MA, se toma el valor de la variable i y se ajusta el valor de manera aleatoria dentro del ancho de banda (b_{rango}). Finalmente se asigna al valor a la variable i de la nueva solución-configuración.
8: **si no**
9: Aleatoriedad (Creatividad). Generar un valor aleatorio dentro del espacio de movimiento del tipo de parámetro y se asigna al valor a la variable i de la nueva solución-configuración.
10: **fin si**
11: **fin si**
12: **fin para**
13: **devolver** x

Como se puede observar, en algunas ocasiones se sigue por el camino que ha dado buenos resultados, en otras se diversifica un poco para salir de óptimos locales, y en otras ocasiones se exploran nuevos espacios siendo creativos. Este es el camino a seguir durante t iteraciones, para así lograr aproximarse o alcanzar la configuración óptima.

4.3.6. Ajuste de tono en un ancho de banda

Sea x el valor actual de un parámetro, li el limite inferior del espacio de movimiento, ls el limite superior del espacio de movimiento, b el ancho de banda de ajuste y x_b el valor del parámetro después del ajuste podrá estar en los siguientes rangos: $li \leq x_b \leq ls, (x - b) \leq x_b \leq (x + b)$.

Ejemplo:

- Parámetro de configuración: Tamaño de la población
- Tipo de dato: Entero
- Espacio de movimiento: [1, 100] (límite inferior y límite superior)
- Ancho de banda de ajuste: 3
- Valor actual del parámetro: 40
- Rango de valores que podrá tomar el parámetro en un ajuste: [37, 43]

Capítulo 5

Procedimiento experimental

5.1. Especificaciones técnicas

- *Sistema Operativo*: MAC OSX "El capitán"
- *Procesador*: Intel core i5
- *Memoria RAM*: 4GB
- *Lenguaje de programación*: JAVA 1.6
- *Compilador*: javac

5.2. Proyecto

A continuación, se explica la funcionalidad de cada clase contenida en el proyecto JAVA. El objetivo es que, en caso de agregar nuevas funciones o seguir con el trabajo de investigación, se entienda la estructura del proyecto. La información se divide por paquetes.

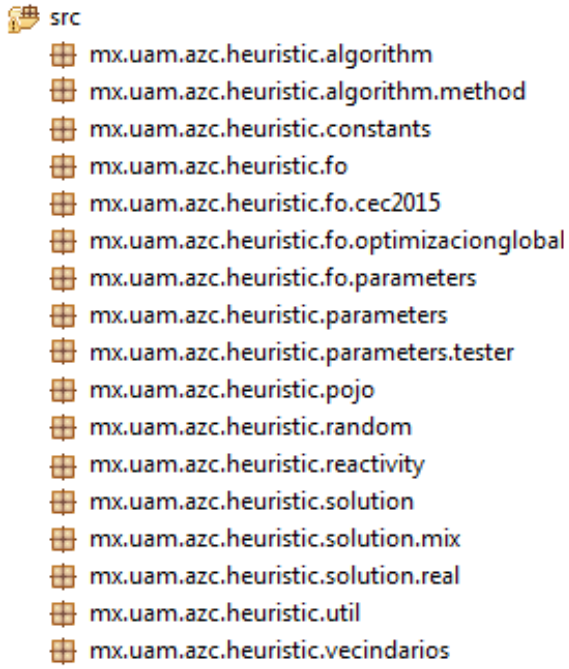

Figura 5-1: Paquetes contenidos dentro del proyecto JAVA

- ***mx.uam.azc.heuristic.algorithm***
  - *HeuristicInterface*: Interface que define los métodos que una metaheurística (CSM o cualquier otra) debe implementar para ejecutarse.
  - *Heuristica*: Clase abstracta que contiene métodos generales como: cálculo del tiempo de ejecución, inicialización de la población, inicialización de las memorias (reactividad). Además, contiene las funciones de vecinadario y el generador de aleatorios.
- ***mx.uam.azc.heuristic.algorithm.method***
  - *CPMS*: Clase que contiene el algoritmo del CSM.
  - *ArmonicaSearchParameters*: Clase que implementa la funcionalidad del calibrador de parámetros por HS.
- ***mx.uam.azc.heuristic.constants***
  - *Constants*: Contiene las constantes globales de la aplicación.
  - *ProblemasCEC*: Contiene valores constantes con repecto a las funciones del CEC como: valor óptimo, número de problema.
- ***mx.uam.azc.heuristic.fo***
  - *FOInterface*: Interface que debe implementar una clase que contenga la lógica de una función a resolver. Por ejemplo la clase *BentCigarFunction* implementa esta interface.
  - *FuncionObjetivo*: Clase padre que se guarda el número de llamadas a la función objetivo, el tipo de problema (minimización o maximización), el óptimo conocido de una función.
  - *OptimizacionGlobal*: Clase que extiende de la super clase *FuncionObjetivo*. Se definen valores como el límite superior e inferior del espacio de búsqueda, así como la precisión al momento de redondear valores reales.
- ***mx.uam.azc.heuristic.fo.cec2015***
  - En este paquete se encuentran las clases obtenidas del CEC 15. Se extraen los métodos que resuelven cada una de las 5 funciones o instancias resueltas en este trabajo.
- ***mx.uam.azc.heuristic.fo.optimizacionglobal***
  - *BentCigarFunction*: Clase que resuelve la función Bent Cigar, extiene de la super clase *OptimizacionGlobal* e implementa la interface *FOInterface*.
  - *DiscusFunction*: Clase que resuelve la función Discus, extiene de la super clase *OptimizacionGlobal* e implementa la interface *FOInterface*.
  - *HappyCatFunction*: Clase que resuelve la función Happy Cat, extiene de la super clase *OptimizacionGlobal* e implementa la interface *FOInterface*.
  - *KatsuuraFunction*: Clase que resuelve la función Katsuura, extiene de la super clase *OptimizacionGlobal* e implementa la interface *FOInterface*.
  - *SchwefelFunction*: Clase que resuelve la función Modified Schwefel, extiene de la super clase *OptimizacionGlobal* e implementa la interface *FOInterface*.
- ***mx.uam.azc.heuristic.fo.parameters***

- *OptimizacionGlobalParameters*: Clase donde se define el número de pruebas y el número máximo de llamadas a la función al ejecutar la metaheurística (CSM) dada una solución-configuración.
- *FOHeuristic*: Clase que implemeta el método para obtener el desempeño de una solución-configuración, extiende de *OptimizacionGlobalParameters*.

- ***mx.uam.azc.heuristic.parameters***
  - *Result*: Clase donde se guada la información estadística de la ejecución del CSM sobre una instancia.
  - *ResultHeuristic*:Clase donde se guarda la información general de una ejecución de la metaheurística, por ejemplo: si encontró el óptimo, número total de llamadas a la F.O, tiempo de ejecución y número de iteraciones.
  - *ResultRealHeuristic*: Engloba la mejor solución y la población al final de la ejecución de la metaheurística.

- ***mx.uam.azc.heuristic.parameters.tester***
  - *Tester*: Clase principal, en donde se ejecuta la metaheurística (CSM) y devuelve como resultado el análisis estadístico.

- ***mx.uam.azc.heuristic.random***
  - *Randomizer*: Clase que contiene métodos para generar aleatorios (enteros, reales, etc.)

- ***mx.uam.azc.heuristic.reactivity***
  - *HeuristicaGoodResults*: Memoria de metaheurísticas generadoras de soluciones. Es decir, lleva el registro de las calificaciones obtenidas por cada metaheurística generadora de soluciones.
  - *Reactivity*: Clase padre que contiene información general que necesitan cada uno de los tres agentes reactivos.
  - *ReactivityInterface*: Interface que define los métodos que debe implementar un agente reactivo.
  - *Reactivityimplementation*: Contenedor de memorias utilizadas por los agentes reactivos.
  - *ReactivityHeuristic*: Contiene la lógica que implementa el agente reactivo que selecciona la metaheurística generadora de soluciones, con base en la experiencia.
  - *ReactivityHeuristicParameters*: Contiene la lógica que implementa el agente reactivo encargado de generar una solución-configuración para la metaheurística generadora de soluciones seleccionada.
  - *ReactivityMove*: Contiene la lógica que implementa el agente reactivo que registra el movimiento de las variables en caso de movimientos perturbantes.

- ***mx.uam.azc.heuristic.solution***
  - *Solution*: Clase padre que registra el costo de una solución y define métodos de comparación con base en el costo.
  - *SolutionInterface*: Interface que define la funcionalidad de una solución.

- ***mx.uam.azc.heuristic.solution.mix***
  - *MixSolution*: Representa una solución-configuración y extiende de la clase padre *Solution*. La solución-configuración puede tener variables de diferente tipo.
  - *MixVariable*: Representa una variable contenida en una solución-configuración.
  - *MixVariableInterface*: Interface que define los métodos que afectan una variable ya sea de tipo real o entera.
  - *BigDecimalMix*: Representa una variable de tipo Real con alto grado de presición.
  - *DoubleMix*: Representa una variable de tipo Real.
  - *IntegerMix*: Representa una variable de tipo Entero.
  - *MixPoblacion*: Engloba un conjunto de soluciones-configuraciones. Es la estructura utilizada como memoria de soluciones-configuraciones en el proceso reactivo.
- ***mx.uam.azc.heuristic.solution.real***
  - *RealSolution*: Estructura de una solución, en donde todas sus variables son de tipo real. Extiende de la clase padre *Solution* e implementa la interface *SolutionInterface*.
  - *RealPoblacion*: Engloba un conjunto de soluciones llamado Población y define métodos de inicialización.
  - *Compositor*: Memoria de un compositor.
  - *Red*: Conjunto de compositores, así como los vínculos entre ellos.
- ***mx.uam.azc.heuristic.util***
  - *TimeExecution*: Clase de utilería, con métodos para cálculo de tiempos de ejecución.
  - *Util*: Clase de utileria con métodos generales que se utilizan constantemente.
- ***mx.uam.azc.heuristic.vecindarios***
  - *VecindariosRealSolution*: Clase en donde se encuentran los métodos de cruza, funciones de búsqueda, etc.

5.3. Entregable

Debido a que el proyecto fue programado sobre la plataforma JAVA, el archivo ejecutable es un JAR de nombre *CSM.jar*, que lee los parámetros de entrada del archivo *input.csv*, ejecuta el algoritmo de CSM sobre una instancia de optimización global continua sin restricciones (el número de veces requeridas) y devuelve los resultados de las ejecuciones, así como el análisis estadístico en el archivo *result.csv*. Un archivo .csv es un archivo de texto plano que puede ser abierto como hoja de cálculo, con la finalidad de tener una forma amigable de introducción y lectura de información.

5.4. Requerimientos para ejecución

Tener instalada una versión de JAVA (6 o superior). JAVA trabaja sobre cualquier sistema operativo (Windows, Linux, Unix, MacOS, Android, Solaris).

5.5. Archivo de entrada *input.csv*

En este archivo se coloca la información de entrada para la ejecución de CSM. La información requerida es la siguiente:

- *Seed*: Semilla para el generador de números aleatorios. Se debe colocar un valor entero. Si no se requiere una semilla específica, colocar "NA".
- *NumFuncion*: Indica la función a probar:
  - Bent Cigar Function (1)
  - Discus Function (2)
  - Schwefel's Function (4)
  - Katsuura Function (5)
  - HappyCat Function (6)
- *Dimension*: Número de variables involucradas en la función. Número entero positivo mayor que 1 y menor o igual que 30.
- *Precision*: Precisión (número de decimales) con la que se manejan los números reales. Número entero positivo menor o igual que 50.
- *MAX-Call-FO*: Límite de llamadas a la función objetivo, que al alcanzarse, se detiene la ejecución i y se sigue con la siguiente ejecución ($1 < i < NumExecutions$).
- *NumExecutions*: Total de ejecuciones de CSM. Cada ejecución se realiza con los mismos parámetros de entrada (n, m, tco, $tpar$).
- *Parámetros de entrada*:
  - Número de compositores (n): Valor entero positivo mayor que 2. Número de compositores que colaboran entre si.
  - Número de melodías por compositor (m). Valor entero positivo mayor que 3. Capacidad de memoria por compositor, es decir, cuántas melodías puede recordar un compositor simultáneamente a lo largo de la ejecución del algoritmo.
  - Tasa de trabajo colaborativo (tco). Valor real entre 0 y 1. Define la probabilidad que los compositores colaboren entre si, o en caso contrario que se comporten de forma aislada o abusiva.
  - Tasa de comportamiento parásito ($tpar$). Valor real entre 0 y 1. Define la probabilidad condicional de que, dado que no se comportan de forma colaborativa, los compositores adopten un comportamiento parásito o, al contrario, aislado.

A continuación, se muestra un ejemplo de cómo introducir los datos de entrada.

	A	B	C	D	E	F	G	H	I	J	K
1	Seed	NumFuncion	Dimension	Precision	MAX-Call-FO	NumExecutions	n	m	tco	tpar	
2	NA	1	30	8	100000	100	3	5	0.1	0.07	
3											

Figura 5-2: *input.csv*

5.6. Ejecución

A continuación, se describen los pasos a seguir para la ejecución:

- Abrir la consola de línea de comandos.
- Posicionarse en la ruta donde se encuentra el archivo *CSM.jar* e *input.csv*. Ambos archivos tienen que estar en la misma ruta.
- Escribir el siguiente comando: java -jar CSM.jar
- El programa crea el archivo de resultados *result.csv*; en caso de ya existir, lo reemplaza.

5.7. Archivo de resultados *result.csv*

El archivo de salida contiene la siguiente información:

- *FUNCTION*: Describe la función con la cual se realizó la ejecución.
- *Parámetros de entrada*: Los mismos parámetros introducidos en el archivo de entrada (n, m, tco, $tpar$).
- *MINIMUM*: Menor número de llamadas a la F.O empleadas en una ejecución para alcanzar el valor óptimo.
- *MAXIMUM*: Mayor número de llamadas a la F.O empleadas en una ejecución para alcanzar el valor óptimo, o cuando se finalizó la ejecución al alcanzar el número máximo de llamadas.
- *AVERAGE*: Promedio de llamadas a la F.O empleadas para alcanzar el valor óptimo, considerando el total de ejecuciones.
- *MEAN*: Mediana.
- *STD*: Desviación Estandard.
- *VARIANCE*: Varianza.
- *VAR-LINF*: Límite inferior de la varianza, proporcionado por prueba bootstrap.
- *VAR-LSUP*: Límite superior de la varianza, proporcionado por prueba bootstrap.
- *MEAN-LINF*: Límite inferior de la mediana, proporcionado por prueba bootstrap.
- *MEAN-LSUP*: Límite superior de la mediana, proporcionado por prueba bootstrap.

Además, se muestra por cada ejecución, el número de llamadas a la F.O, ordenados en orden ascendente.

Capítulo 6

Instancias de prueba

A continuación, se muestran las instancias con las que se probó la eficiencia del CSM. Se muestra la ecuación, la gráfica en 3 dimensiones y sus principales características. Cabe mencionar que las instancias se obtuvieron del CEC'15 (Congress on Evolutionary Computation, 2015) [16]. En todos los casos el óptimo es conocido.

6.1. Bent Cigar Function

$$f_1(x) = x_1^2 + 10^6 \sum_{i=1}^{D} x_i^2 \tag{6.1}$$

Características:

- Unimodal
- No separable

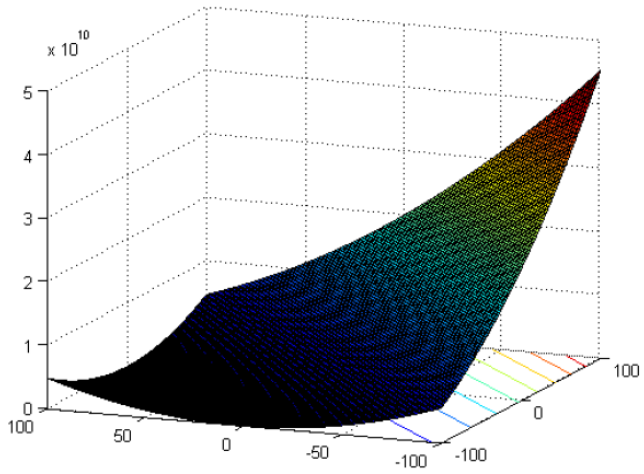

Figura 6-1: Bent Cigar Function

6.2. Discus function

$$f_2(x) = 10^6 x_1^2 + \sum_{i=1}^{D} x_i^2 \tag{6.2}$$

- Unimodal
- No separable
- Con una dirección sensible

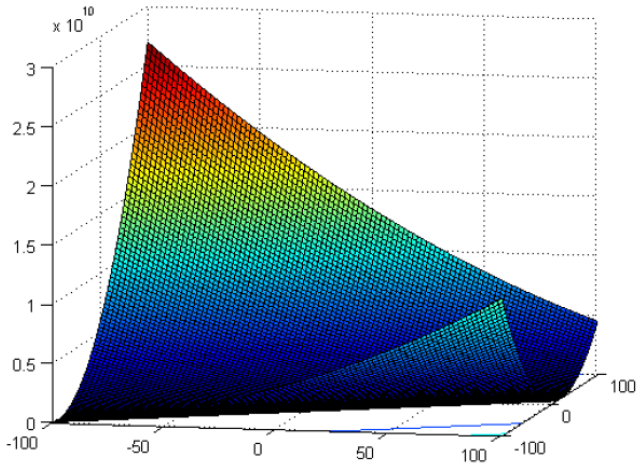

Figura 6-2: Discus function

6.3. Modified Schwefel's Function

$$f_4(x) = 418{,}9829 \times D - \sum_{i=1}^{D} g(z_i)$$
$$z_i = x_i + 4{,}209687462275036 \exp 002$$
$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{\frac{1}{2}}) & si\ |z_i| \leq 500 \\ (500 - \text{ mód } (z_i, 500)) \sin(\sqrt{500 - \text{ mód } (z_i, 500)}) - \frac{(z_i - 500)^2}{10000D} & z_i > 500 \\ (\text{ mód } (|z_i|, 500) - 500) \sin(\sqrt{|\text{ mód } (|z_i|, 500) - 500|}) - \frac{(z_i + 500)^2}{10000D} & z_i < -500 \end{cases} \tag{6.3}$$

- Multi-modal
- No separable
- Gran número de óptimos locales y el segundo mejor óptimo local esta muy lejos del óptimo global

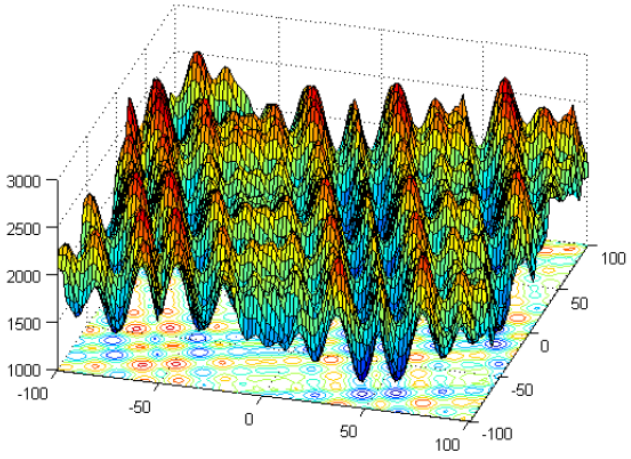

Figura 6-3: Modified Schwefel's Function

6.4. Katsuura Function

$$f_5(x) = \frac{10}{D^2}\Pi_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\frac{|2^j x_i - round(2^j x_i)|}{2^j}\right)^{\frac{10}{D^{1,2}}} - \frac{10}{D^2} \tag{6.4}$$

- Multi-modal
- No separable
- Continua en todas partes, sin embargo, en ninguna parte diferenciable

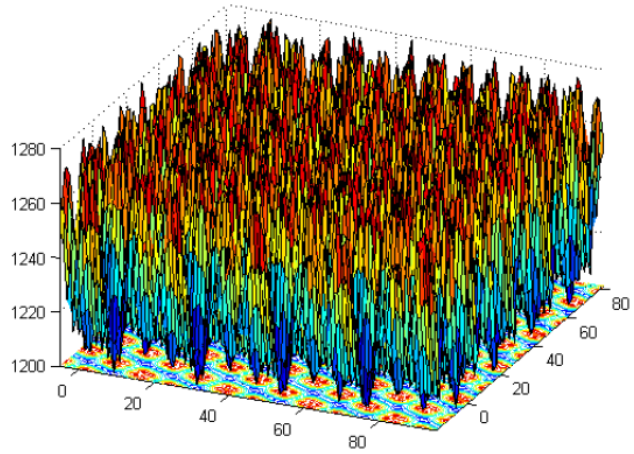

Figura 6-4: Katsuura Function

6.5. HappyCat Function

$$f_6(x) = |\sum_{i=1}^{D} x_i^2 - D|^{\frac{1}{4}} + \frac{0{,}5\sum_{i=1}^{D} x_i^2 - \sum_{i=1}^{D} x_i}{D} + 0{,}5 \tag{6.5}$$

- Multi-modal

- No separable

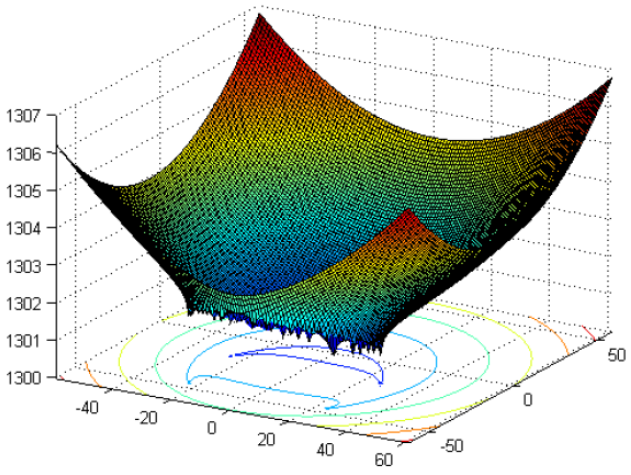

Figura 6-5: HappyCat Function

Capítulo 7

Resultados

En la presente sección se muestran los resultados obtenidos por el CSM sobre un conjunto de instancias de prueba tomadas del CEC'15 (Congress on Evolutionary Computation, 2015) [16]. Como se mencionó en la sección anterior, estas instancias poseen características que dificultan su resolución. Por ejemplo: poseen múltiples óptimos locales, lo cual implica que en el diseño del método se involucran estrategías que le permiten escapar de óptimos locales; algunas son multimodales, otras no son diferenciables, algunas otras son no separables, entre otras características. Para concursar en el CEC, la metaheurística candidata debe de encontrar la solución óptima en 500 o menos llamadas a la F.O para el caso de instancias con D=5 y D=10, y 1500 para instancias con D=30; donde D representa la dimensión del vector solución, es decir, el número de variables involucradas.

A continuación, se listan las instancias de prueba (descritas en la sección anterior), seguidas de el valor óptimo conocido:

- Bent Cigar Function (F1), 100
- Discus Function (F2), 200
- Schwefel's Function (F4), 400
- Katsuura Function (F5), 500
- HappyCat Function (F6), 600

Los valores *tco*, *tpar*, n y m son las variables de entrada de CSM. La calibración de parámetros de entrada se realizó con el método propuesto en la sección 4, basado en Búsqueda Armónica; se realizó para las cinco instancias y para cada dimensión (5, 10 y 30), es decir, se realizarón 15 calibraciones de parámetros en total.

Inicialmente se presenta una sección con resultados estadísticos donde se demuestra que CSM alcanza siempre el valor óptimo conocido, y además, lo encuentra con pocas llamadas a la función objetivo (las requeridas por el CEC). Posteriormente, se presenta una prueba de hipótesis paramétrica para aceptar o rechazar el supuesto planteado por el CEC (máximo de llamadas a la F.O) para las cinco instancias. Y finalmente, con el objetivo de comparar a CSM con otras metaheurísticas, se calcula el error generado por cada una de las instancias con D=10.

7.1. Resultados Estadísticos

Sobre cada una de las instancias de prueba se realizaron 100 ejecuciones aplicando CSM después de ser calibrado; en cada una de las 100 ejecuciones, se reporta el número de llamadas a la función objetivo requeridas para que CSM alcance el valor óptimo (o mejor solución

conocida). Este procedimiento se realiza para cada instancia y para las dimensiones (5, 10 y 30).

Sea $x = (x_1, x_2, x_3, \ldots, x_n)$, donde n =100 (ejecuciones) y x_i corresponde al número de llamadas a la F.O empleadas para llegar al óptimo en la ejecución i, se muestran los siguientes valores estadísticos:

- *Mejor*: mín$(x_1 \ldots x_n)$. Indica el menor número de llamadas a la F.O empleadas para alcanzar el valor óptimo, considerando las 100 ejecuciones.
- *Peor*: máx$(x_1 \ldots x_n)$. Indica el mayor número de llamadas a la F.O empleadas para alcanzar el valor óptimo, considerando las 100 ejecuciones.
- *Promedio*: $\frac{\sum_{i=1}^{100} x_i}{n}$. Indica el promedio de llamadas a la F.O requeridas, considerando las 100 ejecuciones.
- *Mediana*: Considerando que x es un conjunto ordenado , la mediana es el la media aritmética de los valores centrales $\frac{x_{50}+x_{51}}{2}$.
- *STD*: Desviación Estándar. Indica la variación esperada con respecto a la media aritmética (promedio).

En el cuadro 7.1 se muestran los resultados estadísticos obtenidos sobre las instancias de prueba con D=5.

	Parámetros configuración				Resultados Estadísticos				
Función	n	m	tco	tpar	Mejor	Peor	Promedio	Mediana	STD
Bent Cigar Function	3	5	0.2	0.2	33	2052	230.91	125	316.78
Discus Function	7	3	0.58	0.05	44	2456	453.98	258	501.68
Schwefel's Function	5	3	0.19	0.11	21	834	160.55	105	154.69
Katsuura Function	5	3	0.6	0.1	21	1207	98.1	54	145.08
HappyCat Function	5	5	0.05	0.56	41	1316	290.78	203	255.91

Cuadro 7.1: Resultados Estadísticos D=5

La figura 7-1, presenta el diagrama de caja y bigote para las funciones evaluadas con D=5.

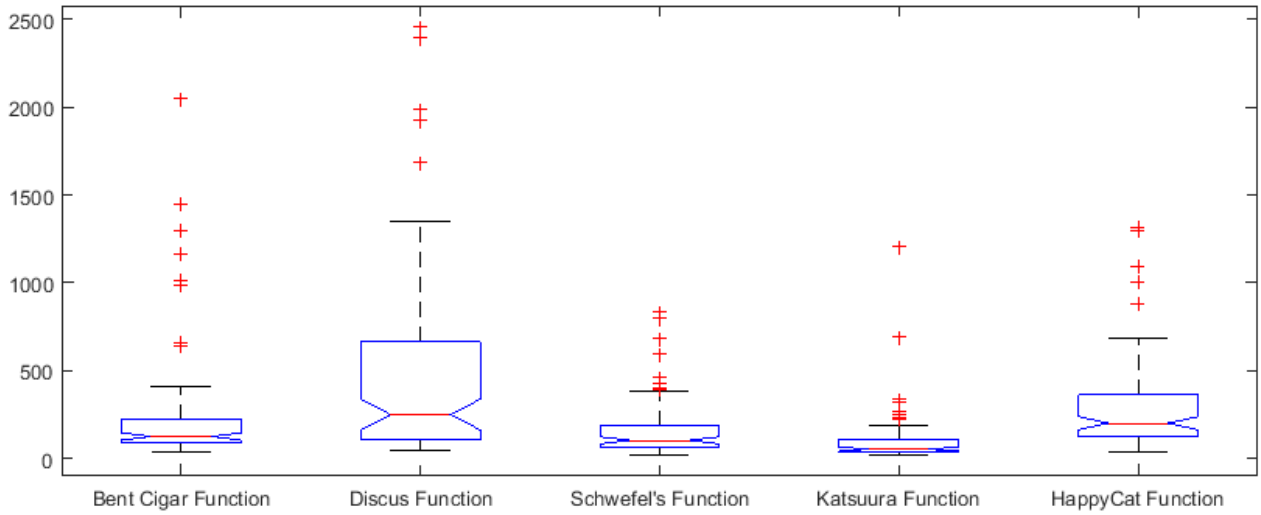

Figura 7-1: Diagrama de caja y bigotes para D=5

En el cuadro 7.2 se muestran los resultados estadísticos obtenidos sobre las instancias de prueba con D=10.

	Parámetros configuración				Resultados Estadísticos				
Función	n	m	tco	tpar	Mejor	Peor	Promedio	Mediana	STD
Bent Cigar Function	3	11	0.04	0.29	66	73823	2765.68	281	8857.28
Discus Function	7	3	0.06	0.51	47	1416	426.2	282	357.51
Schwefel's Function	7	5	0.07	0.63	77	1858	441.19	321	344.78
Katsuura Function	3	9	0.52	0.02	53	1177	205.93	143	186.85
HappyCat Function	7	5	0.02	0.04	59	17948	845.5	176	2470.85

Cuadro 7.2: Resultados Estadísticos D=10

La figura 7-2, presenta el diagrama de caja y bigotes para las funciones evaluadas con D=10.

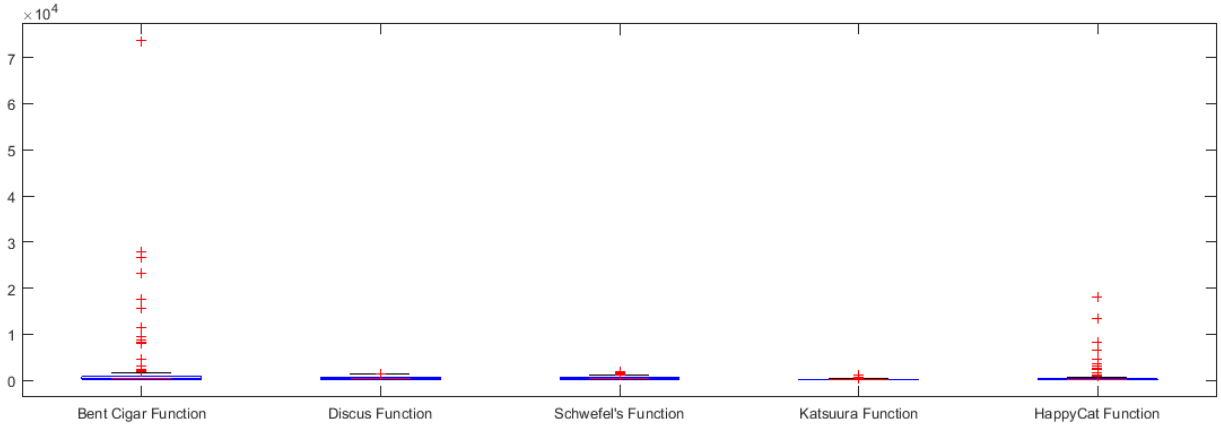

Figura 7-2: Diagrama de caja y bigotes para D=10

En el cuadro 7.3 se muestran los resultados estadísticos obtenidos sobre las instancias de prueba con D=30.

	Parámetros configuración				Resultados Estadísticos				
Función	n	m	tco	tpar	Mejor	Peor	Promedio	Mediana	STD
Bent Cigar Function	21	23	0.23	0.6	620	100089	14627.41	2408	28873.96
Discus Function	5	3	0.2	0.84	54	2435	629.56	563	475.77
Schwefel's Function	3	5	0.04	0.32	30	5847	358	156	678.01
Katsuura Function	5	6	0.12	0.25	42	64008	1941.97	263	7870.98
HappyCat Function	3	5	0.03	0.07	30	29441	1113.93	138	3681.61

Cuadro 7.3: Resultados Estadísticos D=30

La figura 7-3, presenta el diagrama de caja y bigotes para las funciones evaluadas con D=30.

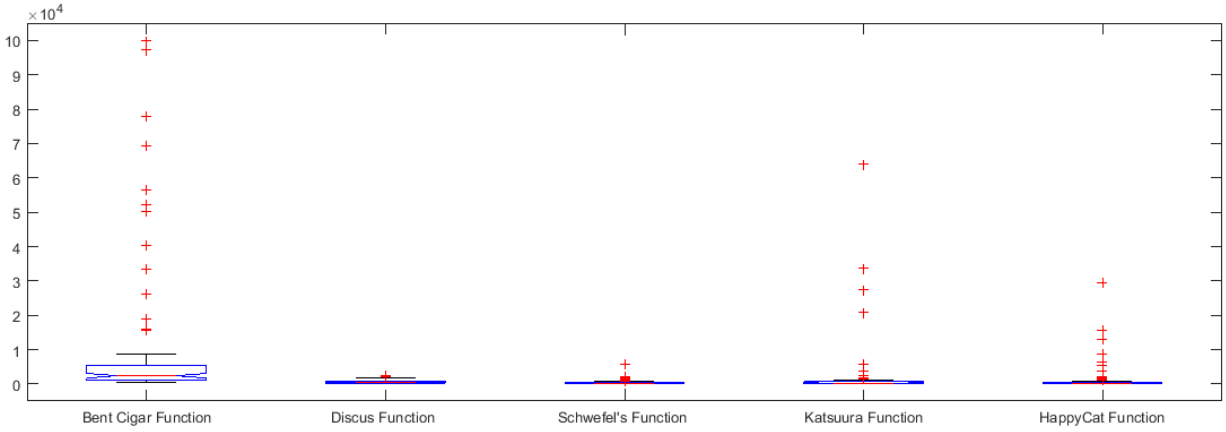

Figura 7-3: Diagrama de caja y bigotes para D=30

7.2. Método de remuestreo Bootstrap

Se realizó el método de remuestreo Bootstrap, el cual se utiliza para aproximar la distribución en el muestreo de un análisis estadístico. Dicho método construye intervalos de confianza que proporcionan con una certeza del 95 % los límites inferior y superior que le tomará al CSM alcanzar el valor óptimo si se ejecuta para la instancia indicada (en base a la mediana). El procedimiento es el siguiente:

- Se toman en cuenta las 100 ejecuciones de CSM para cada instancia, las cuales proporcionan el número de llamadas a la función objetivo que le llevó a CSM alcanzar el óptimo.
- Se seleccionan 20 datos de muestra de forma aleatoria, con reemplazo y se obtiene la mediana. Este proceso se repite 1000 veces.
- Al final se obtienen 1000 datos de mediana. Se ordenan de menor a mayor.
- El límite inferior del intervalo de confianza corresponde al valor en la posición 25 y el límite superior es el valor en la posición 975.

En el cuadro 7.4 se muestran los intervalos de confianza (inferior y superior) obtenidos mediante el método de remuestreo bootstrap para cada instancia con D=5, con una certeza del 95 %.

Función	Límite Inferior	Límite Superior
Bent Cigar Function	94	176
Discus Function	135	660
Schwefel's Function	73	180
Katsuura Function	41	104
HappyCat Function	130	361

Cuadro 7.4: Intervalos de confianza para D=5

En el cuadro 7.5 se muestran los intervalos de confianza (inferior y superior) obtenidos mediante el método de remuestreo bootstrap para cada instancia con D=10, con una certeza del 95 %.

Función	Límite Inferior	Límite Superior
Bent Cigar Function	231	700
Discus Function	180	615
Schwefel's Function	239	572
Katsuura Function	111	215
HappyCat Function	125	319

Cuadro 7.5: Intervalos de confianza para D=10

En el cuadro 7.6 se muestran los intervalos de confianza (inferior y superior) obtenidos mediante el método de remuestreo bootstrap para cada instancia con D=30, con una certeza del 95 %.

Función	Límite Inferior	Límite Superior
Bent Cigar Function	1595	5031
Discus Function	313	855
Schwefel's Function	108	295
Katsuura Function	159	593
HappyCat Function	109	367

Cuadro 7.6: Intervalos de confianza para D=30

7.3. Prueba de hipótesis paramétrica

Con base en los resultados obtenidos (100 ejecuciones por cada instancia), se realizó la prueba de hipótesis paramétrica a través del estaídistico Z. En donde:

- Hipótesis alterna (H_a): Se necesitan a lo mucho 500 llamadas a la F.O para instancias con D=5 y D=10, y a lo mucho 1500 para instancias con D=30.
- Hipótesis nula (H_o): Se necesitan más de 500 llamadas a la F.O para instancias con D=5 y D=10, y más de 1500 para instancias con D=30.
- Nivel de significación: 0.05
- Desviación estándar (STD): Mide la variabilidad dentro de una muestra.
- Error estándar de la media (EEM): Estima la variabilidad entre las medias de las muestras que se obtendría si se tomaran múltiples muestras de la población (datos). $EEM = \frac{STD}{\sqrt{n}}$, donde $n = 100$.
- Estadístico de prueba (Z): $Z = \frac{Promedio-500}{EEM}$ para instancias con D=5 y D=10. $Z = \frac{Promedio-1500}{EEM}$ para instancias con D=30.

El cuadro 7.7 muestra los datos estadísticos empleados en la prueba de hipótesis para instancias con D=5.

Función	Promedio	STD	EEM	Z	Valor de Z al 95	Hipótesis
Bent Cigar Function	230.91	316.7836	31.678	-8.494	1.65	Se acepta H_a
Discus Function	453.98	501.678	50.167	-0.917	1.65	Se acepta H_a
Schwefel's Function	160.55	154.69	15.469	-21.943	1.65	Se acepta H_a
Katsuura Function	98.1	145.075	14.507	-27.702	1.65	Se acepta H_a
HappyCat Function	290.78	255.906	25.590	-8.175	1.65	Se acepta H_a

Cuadro 7.7: Hipótesis paramétrica D=5

El cuadro 7.8 muestra los datos estadísticos empleados en la prueba de hipótesis planteada para instancias con D=10.

Función	Promedio	STD	EEM	Z	Valor de Z al 95	Hipótesis
Bent Cigar Function	2765.68	8857.28	885.728	2.557	1.65	Se rechaza H_a
Discus Function	426.2	357.51	35.751	-2.064	1.65	Se acepta H_a
Schwefel's Function	441.19	344.78	34.478	-1.705	1.65	Se acepta H_a
Katsuura Function	205.93	186.85	18.685	-15.738	1.65	Se acepta H_a
HappyCat Function	845.5	2470.85	247.085	1.398	1.65	Se acepta H_a

Cuadro 7.8: Hipótesis paramétrica D=10

El cuadro 7.9 muestra los datos estadísticos empleados en la prueba de hipótesis planteada para instancias con D=30.

Función	Promedio	STD	EEM	Z	Valor de Z al 95	Hipótesis
Bent Cigar Function	14627.41	28873.96	2887.396	4.546	1.65	Se rechaza H_a
Discus Function	629.56	457.77	47.577	-18.295	1.65	Se acepta H_a
Schwefel's Function	358	678.01	67.801	-16.843	1.65	Se acepta H_a
Katsuura Function	1941.97	7870.98	787.098	0.561	1.65	Se acepta H_a
HappyCat Function	1113.93	3681.61	368.161	-1.048	1.65	Se acepta H_a

Cuadro 7.9: Hipótesis paramétrica D=30

Con base en la información anterior, se puede afirmar que CSM satisface los requisitos del CEC'15 para las 5 instancias con D=5, pero para las instancias con D=10 y D=30 se cumple la hipótesis en 4 de las 5 instancias. Los resultados anteriores muestran que el método propuesto es eficiente en problemas multi-modales y tiene un comportamiento menos favorable en instancias uni-modales.

7.4. Error generado

Con los objetivos caracterizar el comportamiento del CSM e identificar las ventajas y desventajas del método propuesto sobre otras metaheurísticas, se calculó el error generado por cada instancia de dimensión D=10.

Se realizaron 20 ejecuciones por cada una de las instancias, restringiendo cada ejecución a lo más 500 llamadas a la función objetivo. Una vez que se tiene el valor de la función objetivo al alcanzar las 500 llamadas a la F.O, se obtiene el error generado por CSM a través de la ecuación (7.1).

$$error_{i,j} = |f(x^{i,\star}) - f(x^{i,j})| \tag{7.1}$$

Donde: $error_{i,j}$ es el error generado por el CSM en la corrida j de la instancia i; $f(x^{i,\star})$ es el valor óptimo conocido de la función objetivo reportado para la $i-$ésima instancia y $f(x^{i,j})$ es el valor de función objetivo encontrado por CSM para la $i-$ésima instancia en la $j-$ésima corrida.

En el cuadro 7.10 se muestran los valores de los errores generados.

	Schwefel's Function	Katsuura Function	HappyCat Function	Discus Function	Bent Cigar Function
Promedio de error	45.85065	0.0007	0.1935	1.33655	162,539,670.1
Mediana de Error	0	0	0	0	0
Mayor Error registrado	354.269	0.014	0.743	14.246	2,624,033,567
Menor Error registrado	0	0	0	0	0
Desviación Estándar	105.4469228	0.003130495	0.25647889	3.45386742	587,978,041.7

Cuadro 7.10: Caracterización del error. Se muestra el información del error producido por cada función, mediante diferentes valores estadísticos y con respecto a la hipótesis de a lo más 500 llamadas a la F.O.

En las figuras siguientes se muestran los valores de los errores generados por CSM para cada una de las instancias, en diagramas de caja y bigotes.

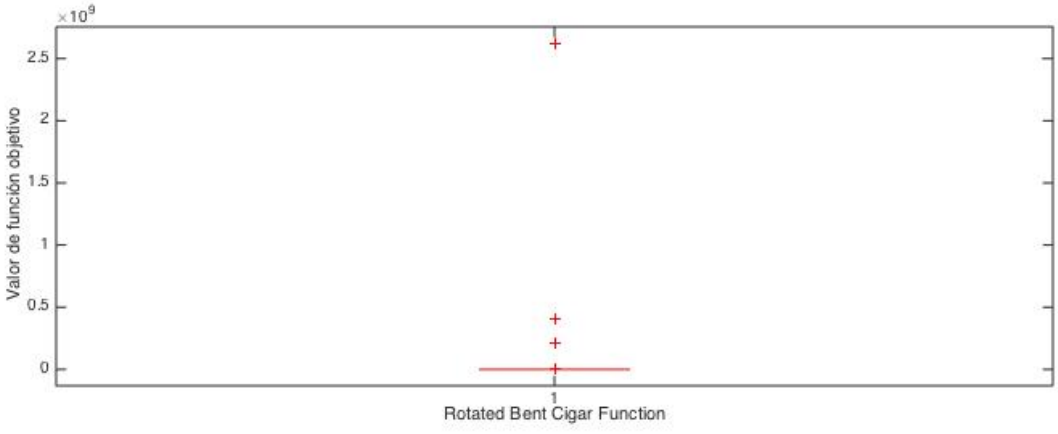

Figura 7-4: Error generado por Bent Cigar Function

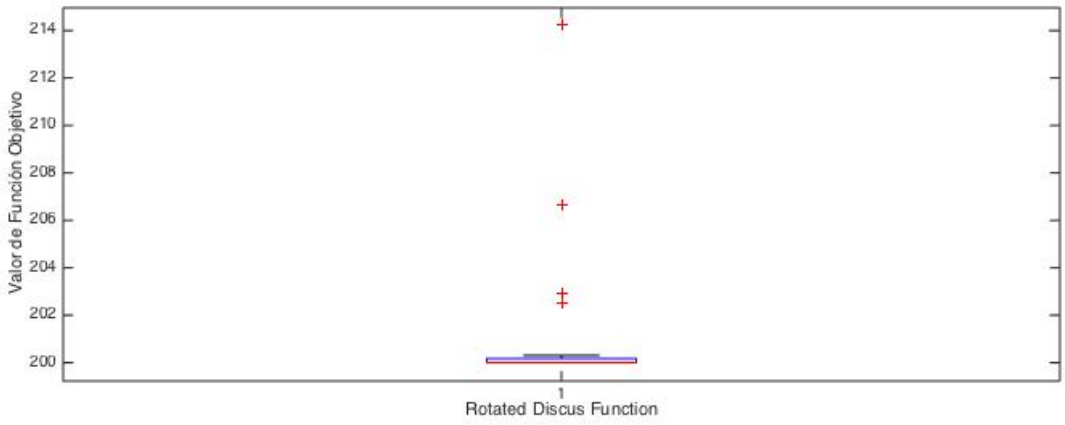

Figura 7-5: Error generado por Discus Function

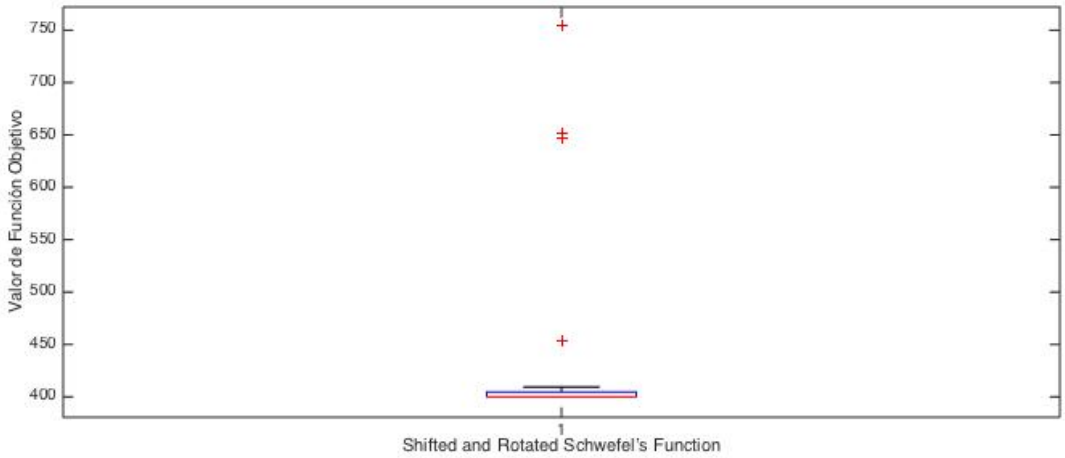

Figura 7-6: Error generado por Schwefel's Function

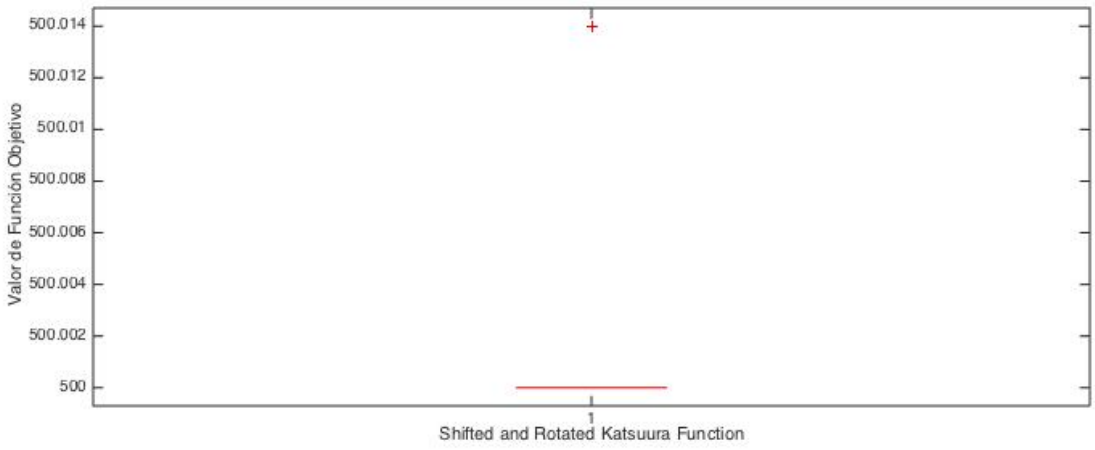

Figura 7-7: Error generado por Katsuura Function

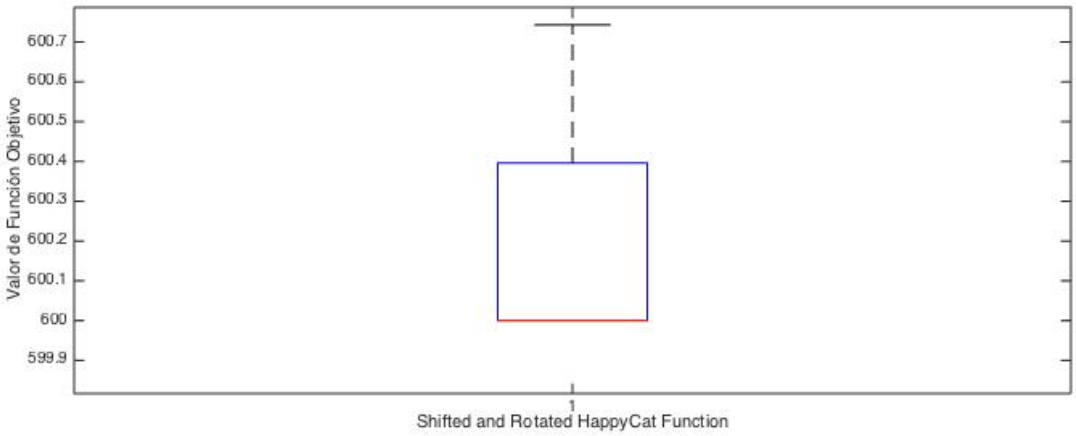

Figura 7-8: Error generado por HappyCat Function

Finalmente, se realizó un estudio comparativo entre los valores de errores generados por el CSM contra los generados por HumangCog (el cual utiliza un paradigma reactivo) [27] (Hc-1), Recocido Simulado [27] (SA), Evolución Diferencial (ED) y Búsqueda Armónica (HS). Cabe mencionar que ED y HS fueron programadas y calibradas con la finalidad de comparar sus resultados con los de CSM.

En los cuadros 7.11 y 7.12 se muestran los valores de error promedio y mediana de error, respectivamente generados por cada uno de los algoritmos.

En el cuadro 7.11 se muestra que en cuatro de cinco instancias CSM produce el promedio de error más pequeño con respecto a las otras metaheurísticas.

	Schwefel's Function	Katsuura Function	HappyCat Function	Discus Function	Bent Cigar Function
CSM	45.850	0.0007	0.1935	1.33655	1.62E+08
HC-1	2.11E+03	2.8	3.5	6.27E+04	3.12E+09
SA	1.23E+03	1.51	4.4	4.14E+04	9.00E+07
ED	889.07	0.993	0.749	5.46E+04	3.00E+08
HS	390.623	0.305	0.807	6.19E+07	5.78E+08

Cuadro 7.11: Promedio de error

En el cuadro 7.12 se muestra que en las cinco instancias CSM produce la mediana de error más pequeña con respecto a todas las metaheurísticas comparadas.

	Schwefel's Function	Katsuura Function	HappyCat Function	Discus Function	Bent Cigar Function
CSM	0	0	0	0	0
HC-1	2.09E+03	2.82E+00	3.63E+00	7.80E+04	3.27E+09
SA	1.30E+03	1.77E+00	4.62E+00	4.55E+04	2.14E+08
ED	869.694	0.984	0.724	1.59E+04	3.07E+08
HS	420.008	0.269	0.815	4.52E+06	5.22E+08

Cuadro 7.12: Mediana de error

Los datos del error (comparando con Hc-1, SA, ED y HS) fueron normalizados y dicha información se muestra en las figuras siguientes:

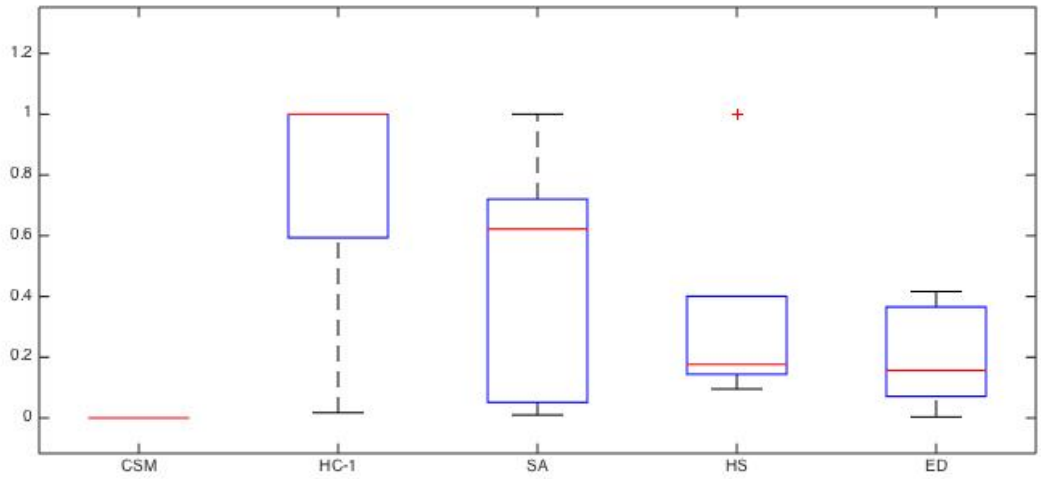

Figura 7-9: Valor normalizado de la mediana del error

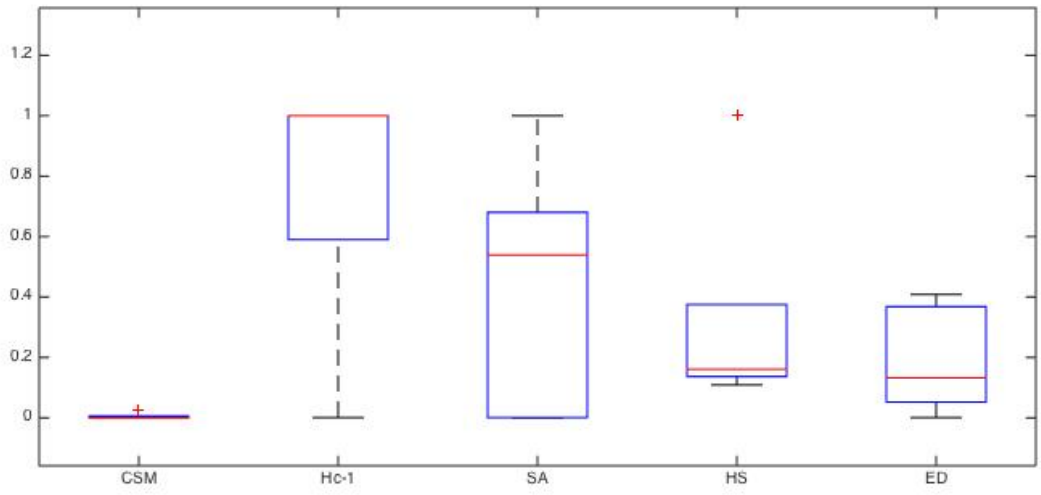

Figura 7-10: Valor normalizado del promedio del error

Con base en la información anterior, se puede afirmar que el CSM es capaz de resolver instancias del problema de optimización global sin restricciones, obteniendo buenos resultados.

Capítulo 8

Conclusiones

El ser humano y el cómo reacciona ante diferentes situaciones o circunstancias en su día a día, siempre ha sido motivo de estudio por parte de los científicos que tratan de entender el por qué de las decisiones que tomamos, y cómo el cerebro realiza el proceso de toma de decisiones. Es claro que intervienen muchos factores como: experiencias previas, miedos, valores, personalidad, entorno de convivencia, moral, temperamento, cultura, etc. Este proceso de toma de decisiones es demasiado complejo, pero existe otra forma de "predecir"la reacción que tendremos ante cierta situación, fijándonos cómo hemos reaccionado con anterioridad en la misma circunstancia. Cada evento se queda guardado en nuestro cerebro, en la "memoria". Entonces, con base en nuestras experiencias previas podemos predecir, con bases probabilísticas, qué reacción tendremos ante una situación que ya hemos experimentado. Si la situación es nueva, queda claro que no se tendrán parámetros de partida y la utilizacin de la experiencia en circunstancias similares se volverá un poco más aleatoria (como cuando se es infante). De las posibles reacciones que se pueden tener, existen algunas que nunca tendremos, simplemente porque no las conocemos o sabemos las consecuencias que pueden tener y optamos por reaccionar como normalmente hacemos o como mejor nos convenga. El ser humano por naturaleza es egoísta y siempre buscara su beneficio o su supervivencia.

Ejemplificando lo mencionado en el párrafo anterior, piense en una persona adulta que nunca ha reaccionado con violencia en su vida en una situación de conflicto. Si a lo largo de su vida, siempre que enfrenta una situación de este tipo, dialoga o se acobarda, lo natural es que elija cualquiera de estas dos opciones, antes de tomar la decisión de reaccionar con violencia. Pero, ¿qué sucede cuando un día reacciona con violencia?. Existe la probabilidad de que gane o que pierda. Si gana se sentirá motivado a reaccionar con violencia en otra ocasión; si pierde se reduce la probabilidad aun más de que lo vuelva a intentar. Del otro lado de la moneda, tenemos al individuo que toda su vida ha arreglado los conflictos con violencia, y un día comienza a experimentar con el dialogo, la negociación o la comprensión. Si estas alterativas funcionan, las comenzará a emplear cuando enfrente una situación de conflicto.

Las experiencias que han tenido ambos individuos a lo largo de su vida, marcaron sus reacciones en eventos de toma de decisiones, pero siempre buscando el beneficio propio o el común (familia, sociedad, grupo de trabajo etc.). Cuando se busca el beneficio común de un grupo de individuos, se debe de tener el conocimiento de qué es bueno y qué es malo para todos. Como los eventos y las reacciones desencadenadas son poco predecibles, puede ser que en otras circunstancias u otros escenarios, una persona que se caracterizó por ser pasiva y tranquila toda su vida, hubiese sido ruda. El éxito alcanzado en su vida no dependió tanto del tipo de decisiones que tomó a lo largo de su vida, sino que el éxito lo obtuvo porque siempre buscó su beneficio y el común.

Haciendo una analogía con el proceso reactivo musical del CSM, la reacción ante la decisión de elegir la mejor técnica para generar nuevas soluciones toma como base las decisiones previas

tomadas por toda la sociedad. Los integrantes de la sociedad de músicos adoptan distintos comportamientos, pero siempre buscan el bien común; el conocimiento es aportado por todos, y todos acceden a él cuando se toma una decisión que pretende mejorar la gama de melodías y encontrar la perfecta armonía (solución óptima). Naturalmente, siempre hay diversos caminos para un mismo fin, y todos los caminos pueden llevar al éxito, lo mismo sucede en la sociedad de músicos. Por ejemplo, en una ejecución del algoritmo las decisiones tomadas para elegir la técnica de generación de soluciones se inclinan hacia las metaheurísticas ED y AG como mejor alternativas, y en otra ejecución ABC y SS son las predominantes . Ambas ejecuciones consiguieron llegar a la solución óptima con el mismo número de llamadas a la función objetivo en promedio. Esto nos indica que realmente no hay un solo camino para encontrar la mejor solución, pero un mecanismo de toma de decisiones reactivo ayuda a que se tomen las mejores decisiones que permiten encontrar la mejor solución en un menor número de pasos. Por otra parte, la sociedad de músicos cuenta con muchas opciones eficaces para realizar la búsqueda, que combinan estrategias de intensificación y diversificación. Cada una de las metaheurísticas internas han sido eficaces resolviendo muchos otros problemas de optimización, y han probado ser excelentes alternativas. Por lo que, CSM tiene métodos de búsqueda eficientes y un mecanismo de elección inteligente, sin casarse con una técnica, ya que a lo largo del algoritmo cualquier técnica puede ser elegida. Esto le da un abanico de opciones en donde si no funciona una opción, otra lo hará. Por ejemplo, puede ser que al principio AG dirija la búsqueda, pero después estancarse en malos resultados (restándole puntos en la matriz de experiencias), dándole oportunidad a ED y SS de guiar la búsqueda y salir de óptimos locales. Como se ha mencionado a lo largo de este trabajo, la toma de decisiones se adapta en base las circunstancias.

En trabajos posteriores se puede extender el programa para resolver problemas con otra estructura de datos, como: Gráficas, flujos, etc; ya que el código está diseñado para cambiar la implementación fácilmente. Se podría crear una nueva versión adaptada a problemas de optimización multi-objetivo realizando una distribución de tareas entre los comportamientos existentes en la sociedad. Otra opción posible, es una extensión para resolver problemas de optimización global con restricciones.

Bibliografía

[1] Campos, P. G., "Desarrollo de una plataforma de optimización global utilizando aritmética por intervalos", Memoria de Título, Ingeniería Civil en Computación e Informática, Dpto. Computación e Informática, Universidad de Tarapacá, Arica, Chile (2004).

[2] D. Karaboga and B.Basturk. A powerfull and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization, 39(3):459-471. 2007

[3] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech- nical repon. Intemational Computer Science Institute, Berkley, 1995.

[4] X.-S. Yang, "Harmony Search as a Metaheuristic Algorithm", in: Music-Inspired Harmony Search Algorithm: Theory and Applications (Editor Z. W. Geem), Studies in Computational Intelligence, Springer Berlin, vol. 191, pp. 1-14 (2009)

[5] Ansgar Bredenfeld and Hans-Ulrich Kobialka, Team Cooperation using Dual Dynamics. GMD - Institute for Autonomous intelligent Systems (AiS), Germany.

[6] Behnke S. y Rojas R. A Hierarchy of Reactive Behaviors Handles Complexity. In: Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From Robocup to Real-World Applications. 2001

[7] L. J. Fogel, A. J. Owens and M. J. Walsh, Artificial Intelligence through Simulated Evolution. John Wiley and Sons, Inc. New York. 1966.

[8] Darwin Charles. The Origin of Species by Means of Natural Selection or The Preservation of Favored Races in the Struggle for Life. The Book League of America. 1929 (publicado originalmente en 1859).

[9] Holland, J.H. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press. 1974.

[10] Kennedy James, Russell C. Eberhart. Particle swarm optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks. 1995

[11] M. Dorigo, V. Maniezzo and A. Colorni, .Ant system: optimization by a colony of cooperating agents,ïn IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 26, no. 1, pp. 29-41, Feb 1996.

[12] Passino, Kevin M. Bacterial foraging optimization. Innovations and Developments of Swarm Intelligence Applications, 2012, p. 219.

[13] Yun Cai. Artificial Fish School Algorithm Applied in a Combinatorial Optimization Problem. I.J. Intelligent Systems and Applications. 2010. pp. 37-43

[14] Martí Rafael, Laguna Manuel, Glover Fred. Principles of scatter search, European Journal of Operational Research, Volume 169, Issue 2, 1 March 2006, Pages 359-372.

[15] Glover Fred , Kochenbergr Gary A. Book of metaheuristics, Kluwer Academic Publishers, 2003.

[16] 2015 IEEE Cogress on Evolutionary Computation (CEC2015). Sandai, Japón. 25 al 28 de Mayo del 2015. http://sites.ieee.org/cec2015/

[17] Mora Gutiérrrez Roman, Ramírez Rodríguez Javier, Rincón García Eric. An optimization algorithm inspired by musical composition. Artificial Intelligence Review. 2012.

[18] Vela Manuel, Cano Benito. Historia filosófica de la sociedad humana,: parte primera. En la imprenta de don Benito Cano. 1797.

[19] Alcántara Almánzar José, Menéndez Alarcón Antonio. Hombre Y Sociedad. Intec. 1987.

[20] Gessler Nicholas. Fostering Creative Emergences in Artificial Cultures. In Artificial Life XII - Proceedings of the Twelfth Internationa Conference on the Synthesis and Simulation of Living Systems.2010.

[21] Christakis, N.A. and Fowler, J.H. Conectados: El Sorprendente Poder de las Redes Sociales y Como Nos Afectan. Aguilar, Altea, Taurus, Alfaguara, S.A. de C.V. 2010.

[22] Edward de Bono. El pensamiento práctico. Editorial Paidos. p 199. 1993.

[23] Bruce L. Jacob. Algorithmic composition as a model of creativity. Organised Sound (Cambridge University Press).pp 157-165. 1996.

[24] Pascual-Mejía Pilar. Didáctica de la Música. Pearson - Prentice Hall. 2006.

[25] Palacios-Sanz José Ignacio. El concepto de musicoterapia a través de la historia. Revista Inteuniversitaria de formación de profesorado. pp 19-31. 2001.

[26] Battiti Roberto, Brunato Mauro y Mascia Franco. Reactive Search and Intelligent Optimization. Dipartimento di Informatica e Telecomunicazioni, Universita di Trento, Italia. 2007.

[27] A. Al-Dujaili, K. Subramanian and S. Suresh, "HumanCog: A cognitive architecture for solving optimization problems,"2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, 2015, pp. 3220-3227.

[28] Glover F., Laguna M., Mart R., Fundamentals of Scatter Search and Path Relinking, Control and Cybernetics, 29 (3), 653-684. 2000.